HUMAN-GUIDED TASK TRANSFER FOR INTERACTIVE ROBOTS

A Dissertation
Presented to
The Academic Faculty

By

Tesca Fitzgerald

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology

August 2020

HUMAN-GUIDED TASK TRANSFER FOR INTERACTIVE ROBOTS

Approved by:

Dr. Ashok Goel, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Andrea Thomaz, Advisor
Department of Electrical and Computer Engineering
The University of Texas at Austin

Dr. Sonia Chernova
School of Interactive Computing
Georgia Institute of Technology

Dr. Henrik Christensen
Department of Computer Science and Engineering
University of California, San Diego

Dr. Brian Scassellati
Department of Computer Science
Yale University

Date Approved: June 24, 2020

To my Mom, who is unstoppable.

Thank you for teaching me by your example and your love.

ACKNOWLEDGEMENTS

I first visited Georgia Tech as an undergraduate, having little idea what a "researcher" was or whether that role could ever be attainable for me. Shortly after, I walked off that campus feeling inspired and in awe by the professors I met who encouraged me to apply to the PhD program, and later, became my thesis advisors: **Ashok Goel** and **Andrea Thomaz**.

As a whole, my PhD journey has often felt like a leap into the unknown. I feel grateful and fortunate to have had **Ashok**'s constant support and wisdom to guide me through it all. Ashok has been my role model in what it means to be a researcher; he taught me to stay curious, to think carefully and critically, and to be bold in trying exciting and unconventional research ideas. And just as importantly, he taught me how to navigate the personal challenges of the PhD: how to bounce back after failures, and how to learn and grow from my mistakes. Thank you for advocating for me and believing in me. I couldn't have asked for a better advisor and mentor.

One of the best reflections of an advisor is the research community they create. I have been fortunate to be a part of **Andrea**'s research lab twice: once at Georgia Tech, and once at UT Austin. In both places, Andrea always saved a place for me, inviting me to join her in some of the brilliant and thoughtful research labs I have ever had the chance to be in. Despite my having never worked in robotics research before the PhD, Andrea took a chance on me and introduced me to the world of robotics. I am grateful for how Andrea continually pushed me as a researcher. She taught me to be clear in my reasoning, careful in my work, and to think deeply through the lens of researchers in various communities. All of these qualities have made me a better, more independent researcher, for which I am deeply grateful. Thank you for investing in me.

I am honored to have been supported by such a brilliant, accomplished committee. **Brian Scassellati** always made the time to talk and invest in my career as a researcher. Thank you for inspiring me to think more broadly about the future of social robots. I

am grateful to **Henrik Christensen** for his support: both directly through his insight as a committee member and mentor to me, and also indirectly as the director of IRIM. Despite the many demands on his time, he always made time to be accessible and ensure that IRIM's graduate students were heard and supported. **Sonia Chernova** adopted me into her lab soon after she arrived at Georgia Tech. I admire her calm, down-to-earth attitude, all while leading an innovative, ambitious research lab. Thank you for mentoring me and guiding me as your own student.

Thank you to the **National Science Foundation**, the **Office of Naval Research**, **IBM**, and **Microsoft Research** for funding my PhD and making my research (and robotics and AI research as a whole) possible.

I am grateful to have been guided by wonderful mentors and role models. **Annie Antón** never hesitated to help me feel welcomed and supported, whether it was helping me talk through a dilemma, or just a coffee check-in to make sure everything was going well. I want to thank **Elaine Short**, who let me adopt her as my unofficial "fourth advisor" and whose invaluable mentorship brought me through the last year of my PhD. I am so grateful for the time I spent learning from you. You made me a better researcher, a better writer, and I hope to one day pay it forward as a mentor for others.

I also want to thank my undergraduate mentors who believed in me and supported me in preparing for grad school. Thank you **Bart Massey** for showing me unending patience, teaching me everything from basic UNIX commands to helping me write and present my first research project. Thank you **Lois Delcambre** for showing me what it meant to be a part of a research lab, and for being involved throughout the PhD application process (even driving me to the airport for my first recruitment visit!).

I am thankful for the wonderful Georgia Tech faculty and staff who looked out for me. **David Kernaghan** went above and beyond his job to make sure I was funded and never missed an administrative deadline. **Cynthia Jordan**, **Josie Giles**, and **Nina Climes** made IRIM run seamlessly and yet still had time to help me with obscure forms, all my

administrative questions, and even making sure the coffee area was well-stocked. I will always be grateful to have met **Jennifer Whitlow** and **Cedric Stallworth** at the Grace Hopper Conference. Thank you for inviting me to visit Georgia Tech, introducing me to my advisors, and making me feel supported and included throughout my time at Tech.

Research doesn't occur in a vacuum, and I was fortunate to learn that while in the company of many amazingly talented and thoughtful colleagues. **Kalesha Bullard** and **Vivian Chu** were my foundation from day one. I could always count on them for insight in whatever crisis I faced; whether it was technical, academic, or personal, they always took the time to listen and search for a solution with me. Thank you for making time for me through many tea-time chats, travel adventures, insightful feedback on last-minute conference papers, and debugging help during midnight robot experiments.

Thank you **Bariş Akgün** and **Crystal Chao** for your patience and mentorship, and for showing me what a successful robotics PhD looks like. **Bryan Wiltgen**, **David Joyner**, **Keith McGreggor**, **Maithilee Kunda**, and **Michael Helms** helped me get my start as a new student; from letting me use their research code to bootstrap my own research work, to having in-depth research discussions, they helped me feel included and at-home in the DILab. I looked forward to my weekly discussions with **Priyam Parashar** and **Snejana Shegheva**, whose enthusiasm and brilliance motivated me to keep moving forward whenever I felt stuck. I am grateful to have been included in the RAIL lab; **Andrew Silva**, **Lakshmi Nair**, and **Weiyu Liu**, thank you for making the lab so welcoming, and for bringing your fresh and exciting research ideas to the lab. **Jon Balloch** and **Angel Daruna**, thank you for the daily banter and procrastination over our cubicle walls that made me look forward to being in the lab every day. **Sid Banerjee**, thank you for your encouragement and camaraderie during all our paper-writing meetups at the coffee shop. **Reza Ahmadzadeh**, **David Kent**, and **Asif Rana**, you never hesitated to help me when I was stuck, whether it was me asking to borrow code or you helping me debug the robot arm for the nth time.

Thank you to the SIM Lab at UT, who fully welcomed and supported me during my time

in Austin. **Alex Gutierrez**, **Ajinkya Jain**, **Akanksha Saran**, **Taylor Kessler Faulkner**, and **Yuchen Cui** were some of the first people I met in Austin. Thank you for making me feel included when I first visited UT, and then making me feel at-home years later when I returned. I could count on **Adam Allevato** to always understand the struggles of finishing the PhD, from hearing me complain "I need to graduate" for the 100th time, to lending his careful critique and advice to my research projects. Thank you **Mai Lee Chang** and **Shih-Yun Lo** for your fresh perspective and letting me be a part of your exciting new research.

I feel so fortunate to have been in the presence of the kind, brilliant community of students I found in RoboGrads, RoboWomen, and GradWomen@CC. I especially want to thank my friends **Amrita Gupta**, **Andrew Price**, **Brian Goldfain**, **Brittney English**, **Carlos Nieto**, **Grady Williams**, **Kayla DesPortes**, **Laura Strickland**, **Nolan Wagner**, **Paul Drews**, **Sasha Lambert**, **Sean McCully**, **Shray Bansal**, and **Shan Tie**. Thank you all for letting me be a part of your cohort and including me in so many wonderful adventures. I could have never anticipated how much being a part of your group would help me learn to be a better researcher and person, and made me want to be a better friend in return. Thank you **Kayla DesPortes** for joining me at coffee shops at 5 AM so we could be writing buddies. Thank you **Laura Strickland** for being one of the most thoughtful people I've ever met, and always being the first to offer help whenever I ran into trouble. Thank you **Andrew Price** for your constant support. Whether it was debate practice during absurdly long bike rides, venting during our shared existential crises, or research conversations over ice cream, you helped me grow as a person during my time at GT.

None of this would have been possible without the support of my family. Thank you to my **Mom (Ami Fitzgerald)** and **Dad (Mark Fitzgerald)** for homeschooling me and raising me to be curious. I am forever grateful for the many sacrifices you made to give me my education. Thank you for teaching me to always challenge myself.

I especially want to thank my **Mom**: I know that my entering the PhD was as much a leap of faith for you as it was for me. Thank you for even entertaining the absurd idea of

your teen daughter moving to the opposite end of the country and, after a lot of negotiating, going on to support me every step of the way. Thank you for worrying about me even when I insisted you didn't need to. Thank you for being the strongest and wisest person I know. Thank you for listening and being invested in me, through all the most celebratory highs and the most despairing lows I encountered during these past few years. You have been my biggest supporter throughout this whole journey, and I am grateful to be even closer to you now than before I left.

With every new experience and challenge I've faced in the past few years, I am grateful to my siblings for being there to figure out life alongside me. Thank you **Tayt Weingarten** for believing in me while also bringing me to my senses, and for all the well-timed phone calls that came right when I needed someone to talk to. Thank you **Tylise Fitzgerald** for reminding me to keep my priorities balanced, and for all the online games and silly videos to distract me when I most needed it. Thank you **Locke Patton** for being a part of my chosen family, and for reminding me that I am never alone.

Finally, this process has challenged me completely as a person, including spiritually. I am thankful for **God**'s guidance through every aspect of this journey; for the mysteries of human cognition that inspired me to study artificial intelligence in the first place, for giving me hope, purpose, and community when I felt lost and uncertain about the future, and for bringing me help and recovery when I felt overwhelmed with the chaos and stress in my life. With every struggle I encounter, He has met me with purpose and unending grace:

> *Therefore, having been justified by faith, we have peace with God through our Lord Jesus Christ, through whom also we have access by faith into this grace in which we stand, and rejoice in hope of the glory of God. And not only that, but we also glory in tribulations, knowing that tribulation produces perseverance; and perseverance, character; and character, hope. Now hope does not disappoint, because the love of God has been poured out in our hearts by the Holy Spirit who was given to us.* Romans 5:1-5 (NKJV)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Adaptability is an essential skill in human cognition, enabling us to draw from our extensive, life-long experiences with various objects and tasks in order to address novel problems. To date, robots do not have this kind of adaptability; yet, as our expectations of robots' interactive and assistive capacity grows, it will be increasingly important for them to adapt to unpredictable environments in a similar manner as humans.

While a robot can be pre-programmed for many tasks and their variations, specifying these behaviors would require tedious effort, and still would not adequately prepare a robot for every scenario it may encounter. This is due to the various dimensions along which the original ("source") and novel ("target") task may differ, such as changes in the task goals, task objects, manipulation tools, task constraints, and task dynamics. Existing work aims to generalize across variations in one of these dimensions by having the robot continue to explore its new environment, or by having a human teacher provide enough demonstrations to cover the space of possible task variations.

Rather than require more demonstration data in order to attempt generalization across these contexts, this dissertation instead leverages continued interaction with the teacher within the context of the target task. This enables a robot to quickly learn the salient task differences for transfer to a specific environment, regardless of the number or variations of previous demonstrations. This first requires an understanding of how task differences, interaction, and transfer are related. This dissertation defines a taxonomy of transfer problems that models the relationship between task context and information requirements for transfer; the difference between source and target problems dictates (i) the level of abstraction at which the task representation should be transferred and (ii) the dimensionality of the information needed to ground that representation for the target problem. Since the mode of interaction between the robot and teacher affects the dimensionality of the learned information, it follows that the mode of interaction should also be selected according to the

level of abstraction at which the task is represented for transfer.

Based on this taxonomy, this dissertation analyzes a particular category of transfer tasks: those in which the source and target environments differ in the objects used to complete the task. The contextual nature of object usage is a primary challenge of this problem; the objects the robot should use (and the order which they are used) is dependent on the task goals, subgoals, and how objects are used to fulfill those goals. This dissertation presents a targeted method of interaction (indicating the next object the robot should use) and corresponding algorithm to infer the object feature that dictates the object mapping within the context of a particular task. This enables the robot to learn to predict the mapping using limited assistance with the first part of the task, and then transfer the remainder of the task autonomously. Furthermore, we identify the effect of noisy feedback during interaction and present a confidence-guided approach to moderating the robot's requests for assistance.

We next consider object replacements that alter the manipulation constraints of the task. To address this category of transfer problems, a different interaction mode is needed to ground the relationship between (1) the new object and (2) the trajectory adaptations necessary to use the new object. This dissertation discusses a higher-dimensional form of interactive assistance, corrections, to record and model constrained points in the robot's motion. Not only do we find that corrections are sufficient for the robot to model the new constraints afforded by the tool within the context of the corrected task, but also that the learned model can also be reused on other tasks that provide a similar context for that tool (e.g. in the tool surfaces used to execute the task).

Overall, this work enables a robot to leverage the teacher's understanding of the task goals and constraints. By identifying (1) multi-modal interaction methods for providing transfer assistance to the robot, (2) the dimensionality of their output, and (3) algorithms for modeling the resulting information provided by the teacher, this work enables a robot to address a wide variety of transfer problems without extensive demonstrations or domain-specific knowledge, and thus contributing toward a future of adaptive, collaborative robots.

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Motivation

Historically, robots have been successfully used in industrial environments in which they complete repetitive tasks in controlled environments. This success relies on two important properties. First, the robot is the only agent in its workspace (enforced by isolating the robots' workspace from humans). Second, the robot's task and environment do not change in unexpected ways (enforced by applying robots only in routine tasks). As robots become more common, we expect that they will also become increasingly common in human environments. Research on social robots has already targeted assistive domains [1, 2, 3, 4] such as home, healthcare, educational, and tourism settings (Figure 1.2). As a result, robots will need to be capable of interaction with humans in a variety of environments.

In contrast to industrial robots (such as those in Figure 1.1), interactive robots will be most useful if they can accommodate the novelty that occurs in human environments. Incorporating robots into human environments introduces several types of novelty. First, robots will need to operate in unstructured environments where humans are entering and exiting frequently, and thus are not expected to constantly supervise the robot. Second, for a robot that interacts with objects in its environment in order to fulfill its tasks, it will need to account for humans relocating, removing, or replacing the objects without the robot's knowledge. Furthermore, human environments contain an endless number of object variations, with examples ranging from the branding and labeling of consumer goods to the dimensions and shapes of handheld tools. Finally, in addition to accommodating unstructured environments, the robot may need to learn new tasks on-the-fly that are tailored to its workspace domain.

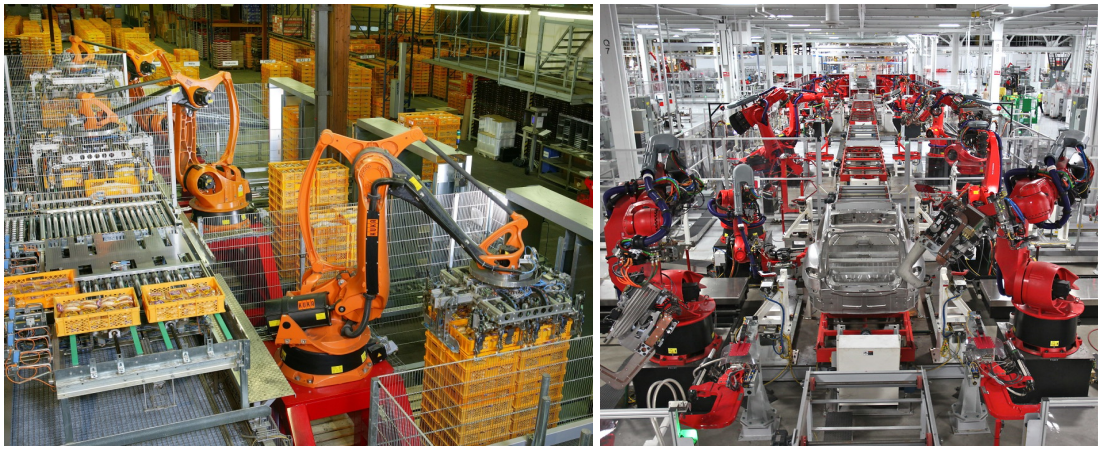


Figure 1.1: Industrial environments in which robots perform repetitive tasks in areas isolated from humans. Images from [5] (left) and [6] (right).

Figure 1.2: Social robots assisting humans in office, airport, and even spacecraft settings. Images from [7, 8, 9].

As a result of operating in human environments, robots will need to adapt to novel environments and objects. Not only is it important that a robot be able to account for this novelty, but it must also account for how different *types* of novelty affect the robot's actions in different ways. Assuming that the robot has learned a set of models for various tasks, these "source" task models will have been learned within the context of their training configurations, defined by the environments, reward functions, and task goals used to train the model. When addressing a new, "target" task, that task's configuration that differ in any of these respects. This dissertation focuses primarily on the effect of changes in the robot's *environment* on task execution, and does not address transferring reward functions or task goals.

Within the scope of addressing environment changes, there are several additional types of novelty that are introduced by object changes or replacements. As shown in Figure 1.3, changes in the location, dimensions, appearance, 3D shape, and/or affordances of a new object must all be addressed for the robot to transfer a task model successfully to the new environment. However, these changes will affect the task in different ways. For example, relocating an object will have a minor effect on task execution compared to replacing one tool object with another (in which case, the task adaptation depends on how the tool is used within the context of the task). As a result, the type of novelty encountered in a target environment affects how a robot should address this novelty.

1.2 Existing Approaches

One method to enable generalization over variations of a task and/or environment is to expand the training set such that it spans the expected task configuration space. This may be effective when the target task configurations are known a priori. However, for a robot in an unstructured, interactive setting, we cannot anticipate every novel situation the robot will encounter, and so expanding the training set accordingly is not practical. Additionally, the training set would need to span the task configuration space across both novel environments

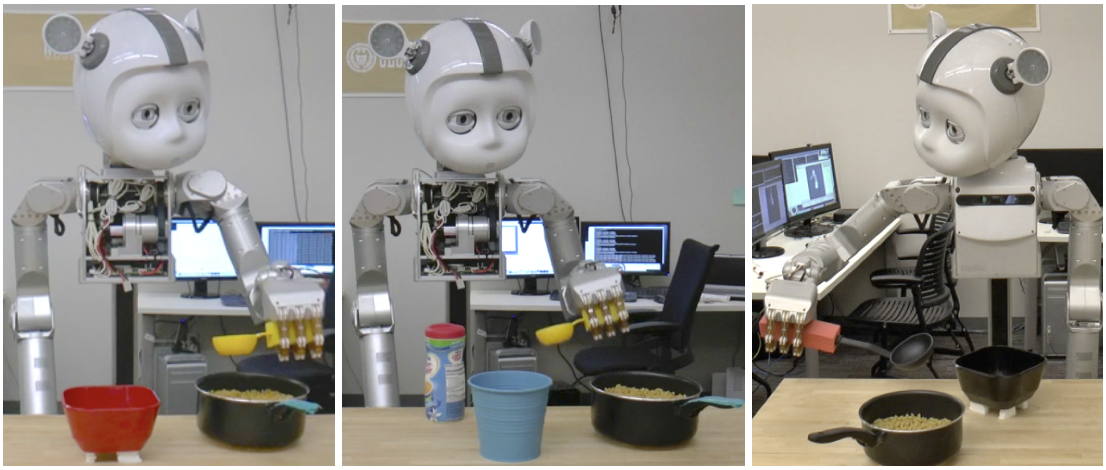


Figure 1.3: Three variations of a scooping task scene. The same task can be completed in all three environments, but the various shapes, appearances, locations, and dimensions of the objects will all affect task execution.

and novel tasks in order to model the relationship between changes in either dimension.

Another plausible approach is to have the robot simply re-learn the task for each task variation it encounters. Since the robot operates in human environments, we expect that the robot may continue to interact with humans and ask for assistance when needed. Learning from Demonstration (LfD) [10, 11] enables a robot to quickly learn from human demonstrations of a task, which a human teacher can provide by moving the robot's arm to complete the task [12, 13, 14] as shown in Figure 1.4. Rather than having a robot learn from training data a priori, LfD enables a robot to receive interactive demonstrations from a human teacher in novel task configurations. LfD provides several benefits:

- It enables the robot to learn a task model that is directly grounded in the target task configuration.
- It enables a human teacher to provide training data to the robot by guiding its actions, rather than programming or otherwise defining the task model manually.
- For kinesthetic LfD, the teacher guides the robot's motion directly, and thus the recorded demonstration is inherently grounded in the robot's own action space (rather than in the teacher's).

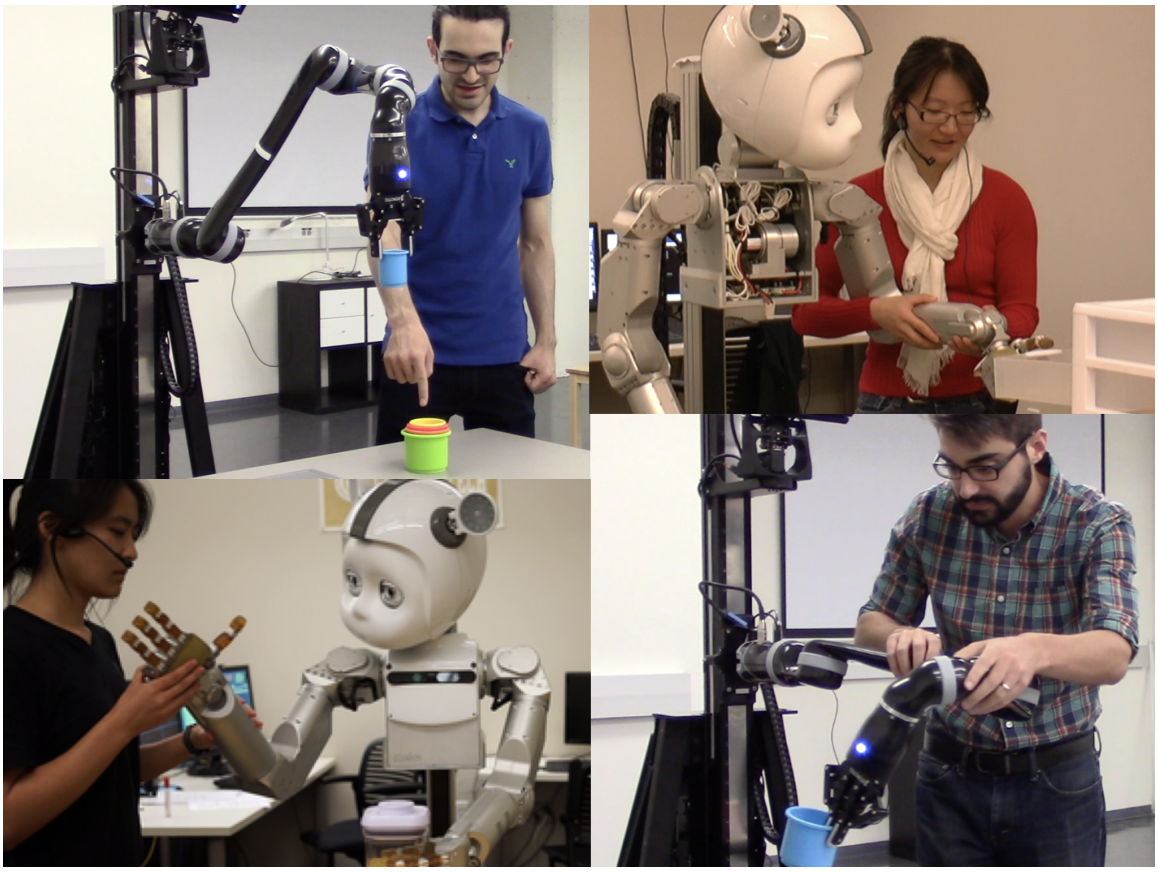


Figure 1.4: Learning from Demonstration enables a robot to record its motion as a human teacher moves its arm to complete a task

- Demonstrations are provided by the teacher based on their understanding of the task configuration. As a result, the teacher can convey their contextual knowledge about the task, such as strategies for efficiently completing the task, or task constraints/requirements that would otherwise be unobservable to the robot.

However, there are also drawbacks to relying solely on demonstrations to address novel task variations. Demonstrations may be noisy, due to a naive teacher's unfamiliarity with the robot's mechanics, limitations, or model training status [14, 15]. The robot may not have access to the same teacher over an extended period of time, and thus would need to adapt to potentially inconsistent training data as well. Depending on the task complexity, demonstrations may also be time-consuming for a teacher to provide, particularly if a new demonstration is needed for every novel task configuration. Additionally, the teacher's availability may be inconsistent if they enter or leave the robot's workspace frequently, and thus we cannot guarantee if or when the robot will have access to new demonstrations. Thus, LfD provides a useful tool for obtaining grounded, task-specific training data, but is not a practical means of addressing *every* target task configuration. Overall, while re-demonstrations would maximize the robot's performance on a new task variation, it minimizes the robot's autonomy and is best suited for environments in which novelty occurs infrequently.

An alternative approach is for the robot to explicitly model the relationship between (i) the difference between the source and target environments and (ii) the necessary adaptations to the task model to enable successful execution in the target environment. Recent research in few-shot learning illustrates the effectiveness of this approach; rather than attempting to collect training examples spanning the entire task configuration space, few-shot learning methods aim to learn the relationship between task changes and model adaptations from a few samples of the target task (e.g. new task dynamics [16, 17], or a new goal state [18]). The resulting model can be reused in additional, unseen task configurations. While few-shot learning approaches have been used successfully in computer vision and

simulated reinforcement learning (RL) domains, they require background training on extensive datasets that exhibit similar relationships between input (e.g. environment image or state space) and expected output (e.g. task parameters or policy) data. These extensive datasets, while plausible for simulated or vision-only tasks, are time-consuming to obtain on a physical robot system.

Active learning provides an additional tool that compliments both LfD and few-shot learning. In a robotics context, active learning involves the robot requesting specific training data from a human teacher, generally using its own learning state to inform its queries. The type of interaction (e.g. label, demonstration, and feature queries) used for active learning affects the type of information that is obtained [19]. The subject of the robot's queries may also occur at multiple levels of abstraction, such as the high-level ordering of actions to complete a task [20] or low-level skills such as grasping [21]. Throughout this dissertation, we consider how the robot may direct its interaction with a human teacher in order to obtain and learn from specific information about a task variation.

1.3 Problem Statement

Demonstration Input and Representation: Suppose that a robot operates in human environments and learns to manipulate objects in its environment to complete tasks. In order for the robot to learn a task, a human teacher manipulates the robot's arm to provide a demonstration of successful task completion in the context of the robot's current environment. During the demonstration, the robot records the trajectory of its motion at several key points of the task. These keyframes can be indicated by the teacher (e.g. via voice commands) or autonomously by detecting key events during the demonstration (e.g. opening or closing the robot's gripper, or coming into contact with an object). This results in a discrete demonstration trajectory $d_i = \langle k_1^i, k_2^i, \ldots, k_n^i \rangle$ containing keyframes $k_1^i \ldots k_n^i$ recorded in environment e_i. The teacher may continue to provide demonstrations (e.g. d_{i+1}) of the task in additional environment variations (e.g. e_{i+1}). As a result, each task

$t \in T$ is learned in one or more environment configurations. We do not address navigation or non-manipulation tasks, and thus represent the robot's environment e solely in terms of the object pointclouds $o \in O$ observable by the robot and their poses p with respect to the robot's pose: $e = \langle (o_1, p_{o_1}), (o_2, p_{o_2}), \ldots, (o_m, p_{o_m}) \rangle$, where each object pointcloud is represented as a set of 6D points ($\langle x, y, z, r, g, b \rangle$).

While the environment representation is originally high-dimensional, we expect that the task is performed with respect to a lower-dimensional representation of the environment. Even then, only a subsection of the lower-dimensional features may be relevant for a specific task. While a task demonstration is performed with respect to a small set of relevant features, this feature set is unknown to the robot. More formally, each demonstration d_i provides a sample of the task:

$$t(e_i) = \left\langle k_1^i(\rho(e_i)), k_2^i(\rho(e_i)), \ldots, k_n^i(\rho(e_i)) \right\rangle$$

Where each keyframe k^i is a linear function over some feature representation $\rho(e_i)$ of the environment e_i.

The definition of ρ is essential, as it defines the set of relevant environment features for the task, and in doing so, reduces the high-dimensional pointcloud input into an abstracted environment representation. This function is also non-linear in nature, as the distance between two object pointclouds is not linearly correlated to their effect on task performance. Additional demonstrations may be provided for the same task, which would result in the samples $k_j^i(\rho(e_{i+1}))$ for each keyframe $j \in [0 \ldots n]$.

Task Changes: After receiving the demonstration(s) for a source task t, suppose that the robot needs to execute the task in a new environment e'. We presume that the learned task has already been determined to be suitable for the new environment, and do not address the task of determining *whether* a learned task model can be performed in a new environment. While the robot has sparse samples of the feature representation $\rho(e_i)$ for the training en-

vironment e_i, these samples are unlikely to generalize to a new environment e' containing different objects than e_i.

Role of Interaction: A teacher can continue to provide demonstrations d_{i+1} of the task in the new environment e'. The new demonstration provides another sample of the task:

$$t(e') = \left\langle k_1^{i+1}(\rho(e')), k_2^{i+1}(\rho(e')), \ldots, k_m^{i+1}(\rho(e')) \right\rangle$$

There are an infinite number of trajectory variations that may result in the successful completion of the task. As a result, the distance between the new demonstration's keyframes $k_1^{i+1} \ldots k_m^{i+1}$ in the new demonstration d_{i+1} and those keyframes from the prior demonstration d_i cannot be solely attributed to the difference between e_i and e'. The keyframe functions and ρ are thus intertwined. In order to model ρ independently of any keyframe function, the robot would need to receive many demonstrations in the context of multiple environments.

We aim to sample ρ more directly by structuring the robot's interaction with the teacher. Rather than passively receive demonstrations, the robot may request targeted interaction at one or more points in its task execution, thus reducing the variance of its keyframe data and enabling the robot to model ρ directly from the teacher's feedback.

Objective: Our goal is to enable a robot to learn a mapping between the source and target environment representations ρ such that it can transfer a learned task model to a new environment. The resulting feature representation is an abstraction of the high-dimensional pointcloud input that the robot observes in its environment. Rather than claim a single, optimal abstraction of this data, we consider several levels of abstraction at which the robot can represent its environment.

We aim to leverage the grounded, contextual knowledge imparted by demonstrations, while also maximizing the robot's autonomy by limiting the amount of assistance required

by the robot. For each level of abstraction at which the robot represents its environment, we define a method of interaction that enables the robot to sample ρ at that level of abstraction. Furthermore, we define a corresponding algorithm to filter and model the data recorded from the interaction.

1.4 Thesis Overview

This dissertation presents several approaches to human-guided transfer, each addressing the challenge of (i) obtaining the relevant information from interaction, while accounting for noisy data prevalent in data from human teachers, and (ii) modeling the data such that the resulting mapping can be used to transfer a learned task model.

Figure 1.5 illustrates the structure of this dissertation. We first define a taxonomy of transfer problems, forming a relationship between (1) the level of abstraction at which a task and environment are represented and (2) the data needed to ground the abstracted representation in a target task domain. Following this, we focus on three abstraction levels defined in this taxonomy. For each abstraction, we present a method of interaction and algorithm for modelling the data recorded from the interaction.

The aim of each of these three approaches is to learn a mapping between the task's state space representations with respect to the source and target environments; we assume that the robot has the same joint configuration and limits in the source and target environments (and thus a constant action space), and that the task contains the same structure in both environments. In this dissertation, we address transfer problems in which the same robot arm is used in the source and target environments, and thus the action space remains unchanged. We consider transfer differences according to two aspects of the robot's state space: the objects in the robot's environment, and the state of the robot's end-effector with respect to its environment. Object changes affect the environmental state space, but may or may not affect the robot's state.

In this dissertation, we present an approach to learning the mapping between state

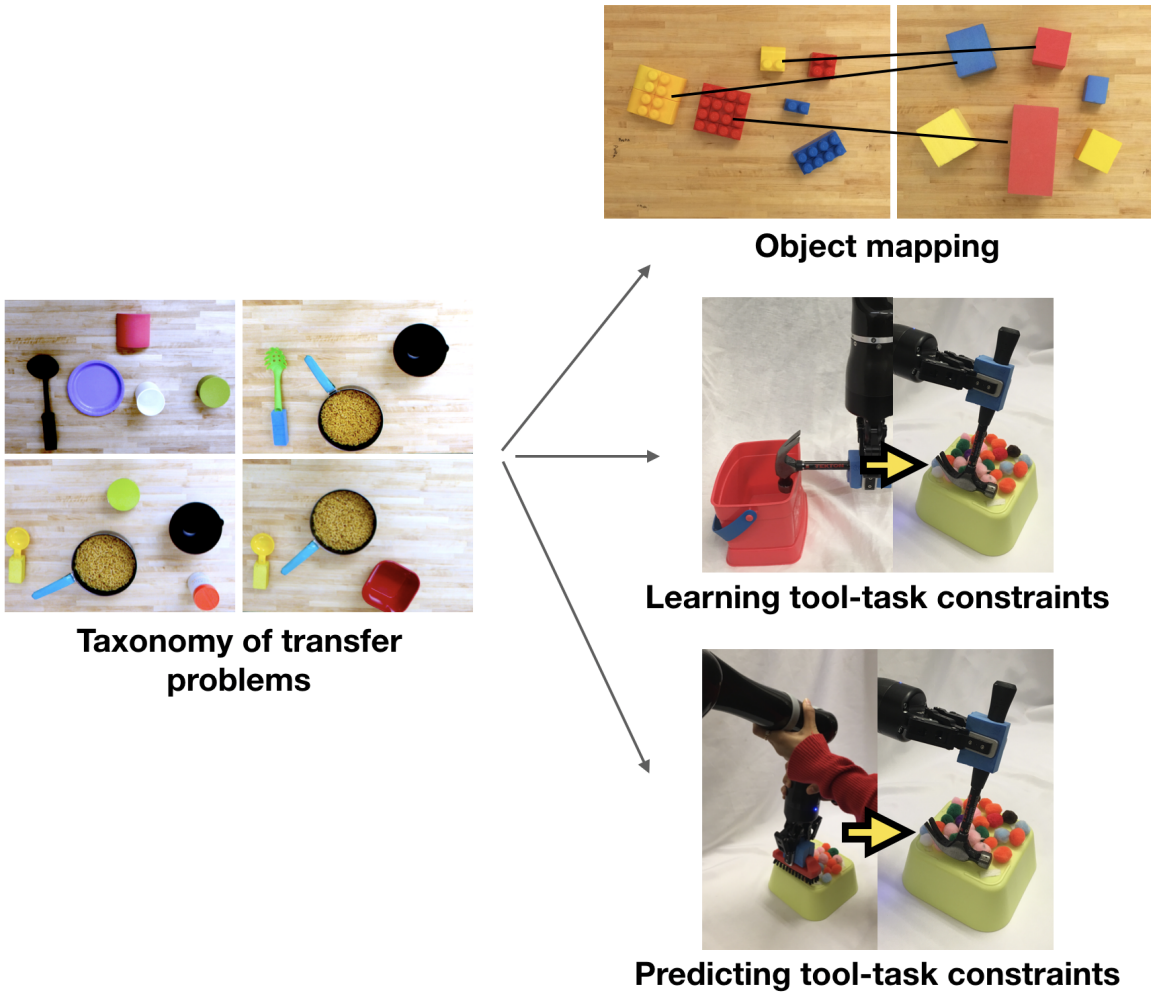


Figure 1.5: Thesis structure: we present a taxonomy of transfer problems, from which, we address three particular categories of task transfer

spaces where:

- Objects *do not* affect the robot state
- Objects *do* affect the robot state and are statically mapped
- Objects *do* affect the robot state and are *parameterized* based on object features

We consider how interaction may be *structured* to provide the data that is needed to learn these mappings. For example, the robot may request a demonstration in a specific environment, or may receive a correction of its motion in the context of a specific task and environment.

1.4.1 Thesis Statement and Contributions

Context: When a robot adapts a known task model to a new environment, there is a relationship between (i) the robot's task model, and (ii) changes in the robot's state space resulting from the new objects and/or tools present in the new environment. Without domain-specific knowledge or extensive task training, this relationship is unknown to the robot.

Thesis Statement: *Learning this relationship from limited training data requires structured interaction between the robot and a human teacher that is tailored to the transfer problem.*

This thesis statement would be falsified if there is a single method of interaction that is sufficient for addressing all transfer problems explored in this dissertation, without requiring extensive training data. We define these transfer problems in our first contribution:

- *A taxonomy of transfer problems* describing the relationship between (i) environment differences, (ii) the level of abstraction at which the task model should be represented, and (iii) the information needed to ground the abstracted task representation in the target environment. When adapting a learned task for a novel environment, the object changes in that environment have an unknown effect on the task execution. For

example, replacing an object in a pick-and-place task affects where the robot should target its actions, but does not necessarily affect the underlying action model. In contrast, replacing a tool that the robot will use to complete a task will effectively alter its end-effector pose with respect to the robot's base coordinate system, and thus the robot's motion must be replanned accordingly. These examples highlight how (i) a mapping is needed between the state space representations of the source and target environments, and (ii) how the mapping's level of abstraction depends on the change that has occurred in the robot's environment, and how that change will affect task execution. We describe how these levels of abstraction align with various interaction modalities in order to ground the abstracted mappings in the target environment. Our experimental results indicate a **trade-off between the generality and data requirements of a task representation, and motivate the need for multiple transfer methods that operate at different levels of abstraction**.

Based on this taxonomy, we make the following contributions, each addressing a different category of transfer problems:

- The *Mapping by Demonstration* algorithm, which this dissertation shows is an effective method for inferring object mappings from interaction with a human teacher. For ordered tasks in which the robot interacts with a series of objects, any replacement of those objects will require a mapping between source and target objects. Without contextual knowledge of the role that objects play in that particular task, the robot is unable to consistently infer the object mapping. This dissertation is the first to address contextual mapping problems using interaction. Furthermore, we investigate the role of noisy feedback from teachers during Mapping by Demonstration, and contribute a confidence-guided approach for self-regulating the robot's requests for assistance. Our experimental data shows that **confidence-guided mapping results in the correct object mapping in over 93% of experiment configurations, while also minimizing the amount of assistance needed from the human teacher**.

- The *Transfer by Correction* method for addressing transfer problems in which an object replacement affects the robot's motion. For example, in tool-based tasks, replacing the tool also changes the tool-tip used to complete the task. We demonstrate how Transfer by Correction enables a robot to learn the mapping between the robot's end-effector state in the source and target tasks via structured interaction. By recording the teacher's corrections of the robot's motion using a new tool, this approach enables the robot to model the relationship between end-effector corrections and the predicted tooltip, despite the correction data containing noise and having a one-to-many relationship to the tooltip position. Our approach succeeded in **modeling these corrections in order to achieve high performance metrics in 83% of tool/task combinations**. Additionally, we demonstrate that after receiving corrections on one task for a new tool, the model learned from those corrections could be reused to transfer additional tasks to that same new tool, **improving task performance in over 27% of transfer problems *without* requiring any additional demonstrations or corrections**.

1.5 Outline of Thesis Document

The remainder of this document is organized as follows:

- **Chapter 2** presents related work, focusing on existing research on transfer learning, analogical reasoning, learning from demonstration, and few-shot learning.

- **Chapter 3** presents a taxonomy of transfer problems. We introduce the Tiered Task Abstraction (TTA): a task representation that can be abstracted to address a range of task differences. We present experimental results demonstrating how the abstracted (and subsequently grounded) representation enables successful transfer in two pick-and-place tasks. This experiment relies on manually-grounded task information, and motivates the need for the robot to ground this information itself via interaction with

a human teacher.

- **Chapter 4** presents Mapping by Demonstration, an interactive method for grounding pick-and-place tasks in a set of unfamiliar objects. We present three sets of experimental results: an extensive evaluation in simulation, a user study in which human teachers provide mapping assistance, and a follow-up analysis of confidence metrics to enable the robot to moderate its own interaction with the teacher.

- **Chapter 5** presents Transfer by Correction, an interactive method for re-parameterizing tool-based tasks when using a new tool object. We show how correcting the robot's motion when using a new tool enables the robot to directly model the transform between its end-effector and the new tooltip. We consider the constraints that tools impose on task execution, and present two models of these constraints. We demonstrate how these models successfully encode the tooltip pose for new tool replacements, and a metric for selecting the constraint model best suited for a particular tool.

- **Chapter 6** proposes a method for learning a *parameterized* tool transform model in order to predict the tooltip for unseen tools. We introduce two primary challenges of this problem: accounting for multiple plausible tooltips, and grounding the visual tooltip prediction into a kinematic transform. We conclude by proposing an evaluation to be completed in future work.

- **Chapter 7** concludes this dissertation and summarizes the contributions of the thesis as a whole. We also discuss open questions and opportunities for future work.

CHAPTER 2
RELATED WORK

We summarize four categories of related work:

- **Interactive learning:** We discuss methods that enable a robot to quickly learn new information by interacting with a human teacher. We summarize interaction methods as well as methods for generalizing models learned from few demonstrations.

- **Solving transfer and generalization problems:** We consider transfer methods that operate independent of human interaction, instead reasoning over past demonstrations to generalize to a new task. We focus on "lazy-learning" approaches that are intended to operate over limited training data, similar to how a robot may need to generalize a task model to a new environment shortly after learning it.

- **Analogical and Case-based Reasoning:** Rather than attempt to generalize over a large set of transfer problems, these approaches aim to identify the relationship between two *particular* source and target problems.

- **Task grounding and mapping:** The high-dimensionality of robot perception and action presents a challenge when modeling the relationship between two environments. We summarize two approaches to this problem: grounding a high-level task representation in a robot's action and perception, and identifying a mapping between two low-level environment representations.

2.1 Learning from Demonstration

Learning from Demonstration (LfD) enables a robot to quickly receive new demonstrations [10, 11] by having the teacher physically guide it to complete a task. This provides

several advantages for a robot that operates in open-ended human environments. First, it enables end-users to "program" a new task for the robot without requiring that they have knowledge of the robot's kinematics, programming language, or learning models. Furthermore, this enables a robot to learn new tasks on-the-fly rather than requiring all task models to be programmed entirely prior to deployment. LfD typically consists of two stages: (1) recording the interactive demonstration and (2) modeling the recorded data.

2.1.1 Interaction Methods for Task Learning

A robot may passively observe a human teacher completing a task (e.g. shadowing the human's actions), and then learn the task goal or even the motion trajectory from this demonstration. However, these passive methods of demonstration may present a *correspondence problem* in which the human's actions do not directly translate to the robot's capabilities (due to the range, location, and number of actuators the robot possesses) [10]. In this dissertation, we primarily utilize *kinesthetic* demonstrations in which the teacher physically moves the robot's arm and/or end-effector to complete the task. As a result, the demonstration is directly recorded in the robot's own action space [12].

Aside from selecting the mode of interaction, the demonstration data can also be affected by the frequency and timing of datapoint recordings. The full demonstration can be recorded at frequent intervals, producing a dense trajectory recording of the demonstration. Alternatively, selected "keyframes" may be recorded around points of interest within the demonstrated trajectory. By reducing the number of recorded points, keyframe demonstrations may provide greater emphasis on constrained parts of the task, while also reducing the likelihood that noisy datapoints will be recorded (such as when a teacher needs to re-record part of a demonstration, or mistakenly demonstrates an action that they do not intend to have recorded) [13, 14]. This dissertation primarily uses keyframe demonstrations as input to our algorithms. The timing of these keyframes may also be defined by human interaction (e.g. via voice commands) or based on the robot's past experience. Interactive *corrections*

have been shown to be effective interface for adapting a previously-learned task model [22, 23, 24] by recording keyframes that most closely align with those of prior demonstrations or task models.

2.1.2 Modeling and Generalizing from Demonstrations

While this dissertation utilizes various methods of interaction to obtain demonstration, we primarily focus on modeling and transferring the data collected from demonstrations. Dynamic Movement Primitives (DMPs) [25] provide a dynamical system approach to task generalization, in which the demonstration trajectory is represented as a perturbed spring-damper system. This model can be parameterized according to the location of the robot's end-effector at the beginning and end of a primitive action [26]. Pastor et al. [27] demonstrate how this model enables generalization to spatial perturbations of the learned task. Demonstrations are segmented into a series of primitive actions, such that only the current end-effector state and the goal state must be specified for a new trajectory to be planned. To repeat an object-centered task in an environment in which the locations of objects have been moved, the robot needs only to re-parameterize each primitive action with respect to the new locations of objects. While DMPs enable generalization from a single task demonstration, it does not represent the relationship between non-spatial changes in the robot's environment (such as changes in object features) and the corresponding model adaptation.

Gaussian Mixture Models (GMMs) can be used to represent variations across multiple demonstrations at their corresponding datapoints, with this variance measured with respect to a feature vector defined by observations of the robot's environment [28]. This model thus reflects its uncertainty when generalizing to new observations. GMMs exemplify the learning paradigm in which the robot's ability to generalize a task model is dependent on its training dataset containing a similar distribution of parameters as those of the target environments.

More recent work has leveraged training data across multiple tasks in order to reduce

the training data needed to generalize to a new task (or new task variation). Bruno, Calinon, & Caldwell [29] address task transfer where (i) the initial set of demonstrations relevant to the target task is limited, and (ii) the relationship between task constraints and the demonstrated actions is unknown. Their method aims to learn both a task model and the parameters of the task. Their approach involves separating *trajectory parameters*, which are used to model the skill common to all demonstrations, from *task parameters*, which reflect the state in which the robot is to repeat the task. Task parameters may include the starting and ending positions of the robot in a demonstration. A Gaussian Mixture Model encodes each task parameter value, and is then used to fit the demonstration to a spring-damper dynamical system representation. They introduce an exploration step which then generates new instances of trajectories based on this dynamical system, each generated under different task parameter values. This increases the size of the demonstration set, and thus the accuracy of the model by enabling sampling from a distribution of demonstrated task parameters, without requiring the user to provide additional demonstrations.

2.1.3 Active Learning

Active learning in human-robot interaction involves the robot selectively requesting training data from a human teacher using its own learning state to inform its queries. Assuming that a robot has access to limited demonstration data, enabling the robot to request demonstrations in particular tasks or environments could result in improved generalizability of the learned model. The type of interaction (e.g. label, demonstration, and feature queries) used for active learning affects the type of information that is obtained [19]. The subject of the robot's queries may also occur at multiple levels of abstraction, such as the high-level ordering of actions to complete a task [20] or low-level skills such as grasping [21]. Maeda et al. [30] present an active learning method for determining when a learned DMP model can be generalized to a new scenario, based on the total variance over each trajectory point returned by a Gaussian Process.

2.1.4 Summary: Limitations of Existing Work in LfD

To summarize this section, interaction enables a robot to quickly model task trajectories, constraints, or modifications. However, demonstrations are not sufficient on their own to enable a robot to generalize to unforeseen task differences as they do not encode the relationship between environment changes and task model modifications. Enabling a robot to request demonstrations in specific task/environment contexts (e.g. using active learning methods) may improve the robot's ability to generalize from this limited training data. Incorporating an active learning approach to task transfer prompts additional research questions, such as how the robot should represent the task or model the similarity between tasks/environments. In the next section, we consider existing research within the transfer learning literature that focuses on these challenges.

2.2 Transfer Learning and Generalization

The aim of *transfer learning* is to use knowledge of a source task to improve the agent's performance in completing a related, target task. Improved performance may be measured according to metrics such as better initial performance in the target task, increased learning speed in the target task, or fewer training instances of the target task [31, 32]. The aspects of the tasks that may differ between the source and target domains (such as initial and final states, action space, and state space) are referred to as *task differences*.

2.2.1 Task Differences

Taylor and Stone provide a breakdown of transfer learning approaches for RL domains according to the *task differences* they are equipped to address [31]. Hayes et. al [33] studies performance differences between a transfer system trained over a training data for a single assembly prediction task and a system trained over multiple source tasks (all various assembly prediction tasks). They show that for the system trained in one assembly prediction

task, its performance is significantly improved when the source and target tasks share similar tool requirements and motor skills. Task differences may also be apparent only after the robot has failed to complete a task; after which, meta-reasoning may be used to address that particular task failure [34, 35]. We explore this notion of task similarity further in Chapter 3, proposing a similarity-based approach to abstracting task representations for transfer.

2.2.2 One/Few-shot Learning

Recent work in one-shot learning has focused on learning a non-linear relationship between environment features and their effect on the task parameters across multiple tasks, such that this model can be quickly tuned for a new task. This may involve learning a latent space for the task in order to account for new robot dynamics [36] or new task dynamics [16, 17]. "Meta-learning" approaches have succeeded at reusing visuomotor task policies learned from previous demonstrations [37] and using a new goal state to condition a learned task network for new task objects [18]. Model-Agnostic Meta Learning (MAML) [38] addresses the few-shot learning problem by learning a non-linear relationship between the task parameters and the model parameters. Clavera et al. [39] adapt the MAML algorithm to learn a policy over an ensemble of dynamics models, enabling fast adaptation for new task dynamics. Hu et al. [40] present a method for addressing a similar problem of disentangling the task embedding and environment embedding to transfer policies across environments and tasks.

Additionally, the relationship between a new object and task constraints can be learned via exploration; Sinapov and Stoytchev [41] present a method for learning from observing the effect of various behaviors (e.g. pushing, pulling, rotating) on an object. Fang et al. [42] use simulated tool models to train a task model over a variety of tool shapes in order to adapt quickly to unseen tools. Zhu et al. [43] present a method using a limited number of demonstrations to leverage pre-learned visuomotor policies for zero-shot transfer from simulation.

In this dissertation, we address a challenge similar to one-shot learning, in that the robot must generalize across a class of problems using limited training data within that class. However, the problem we address differs in that the ground truth is available only via interaction with a human teacher (since there are no datasets that ground the robot's perceptual data in the context of its task execution). Additionally, this ground truth is contextually dependent on the task being performed, which is known by the human teacher but not by the robot. As a result, training data is sparse and subject to noise resulting from the interaction. We next consider existing research that is particularly suited for transfer using sparse data.

2.2.3 Locally Weighted Learning

Locally weighted learning (LWL) operates by combining training data according to their respective distance from the target problem [44, 45]. All training datapoints that are within a thresholded distance from the target problem are retrieved and used to form a local model according to a weighted linear combination function. The definition of this linear combination function is essential in determining how the model should generalize over datapoints. This local model is finally used to predict the outcome of a particular action in that state, with the selected action bearing the highest predicted reward. A primary benefit of the LWL approach is that it avoids *negative interference*, which occurs when adding datapoints to a parameterized model reduces the effect of "useful" data [44, 45]. By storing all datapoints individually, negative interference is avoided. Schaal & Atkeson [46] use a LWL approach for robot control in a juggling task, in which the robot's next command vector is estimated by finding the next *setpoint*, a datapoint whose outcome is predictable with the smallest local confidence interval. This setpoint is then used to retrieve a local linear model, which is then used to construct an linear-quadratic controller appropriately. The setpoint is then shifted again toward the goal position, and the process repeats until the robot's state is within a threshold of the goal state.

When using LWL in high dimensional state spaces, such as those encountered by a robot, it is important to properly weight and normalize the feature vector dimensions. For a newly learned task, however, the robot may not have a model of the prior distribution of feature values, and cannot gain this distribution without a large set of training examples. Additionally, learning the linear combination function would also require a large training set in order to reflect the relationship between task differences and their effect on the model output.

2.2.4 Summary: Limitations of Existing Work in Transfer Learning and Generalization

The difference between the robot's known, "source" data and the new, "target" problem directly affects the data and method used for transfer. Few/one-shot learning aims to learn this relationship, but relies on training data being readily available, and thus is not suited for the sparsity of data collected in an interactive setting. Additionally, the task goals may dictate how the differences between the source and target tasks and/or environments effect the task completion. We address the problem of a robot that has *not yet* been able to explore the effect of these differences on the task goals, and thus needs to learn a task-specific relationship between perceptual changes and the robot's action models.

2.3 Analogical and Case-based Reasoning

Rather than attempt to learn a single model that is generalizable across multiple problems, we now consider cognitively-inspired approaches that aim to identify the relationships between *specific* problem-solution pairs. As a result, these approaches do not typically involve training a generative model over a set of data, but instead perform an analysis over the relationships embedded within and between problem-solution pairs. We now consider how these approaches apply to the problem of task transfer.

2.3.1 Analogical Reasoning

Analogical reasoning is a cognitively-inspired methodology for applying past experiences to a new, unfamiliar problem [47, 48, 49]. A set of *source cases*, each containing an instance of a problem-solution pair, is stored in a *source case memory*. An unfamiliar problem is then addressed using the following methodology. As new problems are viewed, the single, most-similar observed case is pulled from the source case memory. The retrieved case is then compared to the new, *target case*, and a mapping is derived that contains the differences between the two. Using this mapping, the retrieved source case solution is then adapted to address the differences between the two cases. The adapted solution is then deployed in the context of the target case.

The source cases and target problems in analogical reasoning lie on a *similarity spectrum* [50]. At one end of the spectrum, the target case is identical to a source case, and thus can be transferred to the target problem without any adaptation. At the other end of the spectrum, the target problem is completely dissimilar to all source cases available in the case memory and thus cannot be addressed via transfer. In between the two extremes, transfer entails problem abstraction where the level of abstraction may depend on the degree of similarity between the source and target problems [51]. We further discuss the application of similarity to Learning from Demonstration (LfD) in Chapter 3.

Previous work by Floyd et al. has applied analogical reasoning to LfD in the context of robot agents that transfer strategies learned from observing demonstrations of other agents in the RoboCup domain [52]. Each case encodes the behavior and perception of the observed agent at a particular moment, the behavior at which is then transferred when the learning agent perceives a similar situation. In contrast, we identify different degrees of similarity in LfD, and develop methods that operate at different levels of abstraction to address problems at different degrees of similarity.

2.3.2 Case-based Reasoning

Case-based reasoning (CBR) provides a cognitively-inspired account for reasoning and learning in which similar problems are assumed to have similar solutions [47]. Experiences are stored individually as source cases in a *source case memory*, and are retrieved and adapted to address an unfamiliar problem (the *target* problem) by (1) retriving the most relevant source case from memory, (2) creating a mapping that outlines the task differences between the source and target cases, (3) adapting and deploying the source case according to the mapping, (4) evaluating the transfered case's performance, and (5) saving the revised case as a new source case for later usage [47, 53].

Fauconnier & Turner [54] introduced *conceptual blending*: a tool for addressing analogical reasoning and creativity problems, obtaining a creative result by merging two or more concepts to produce a new solution to a problem. Abstraction is enabled by mapping the merged concepts to a *generic space*, which is then grounded in the *blend space* by selecting aspects of either input solution to address each part of the problem. Applied to a robotic agent that uses this creative process to approach a new transfer problem, the robot may combine aspects of several learned tasks to produce a new behavior.

Case-based reasoning has been used to address the problem of transfer in robotics domains. Olteţeanu & Falomir [55] describe a method for object replacement, enabling creative improvisation when the original object for a task is unavailable. Ontanon et al. [56] describe their approach to observational learning for agents in real-time strategy games. They use a case-based approach to online planning, in which agents adapt action plans which are observed from game logs of expert demonstrations. While the domain of strategy games is relevant to our work, games do not present the same challenges of action and perception as a physical robot.

Park & Howard [57] describe a case-based approach to turn-taking in an interactive therapeutic robot, selecting a case that enables the robot to select an object strategy for real-time interaction with the human. Floyd, Esfandiari & Lam [52] describe a CBR approach to

learning strategies for RoboCup soccer by observing spatially distributed soccer team plays. More recently, Floyd and Esfandiari [58] describe an approach for case-based learning by observation in which strategy-level domain-independent knowledge is separated from low-level, domain-dependent information such as the sensors and effectors on a physical robot. Floyd et al. [52] also describes a case-based approach to learning from observing RoboCup soccer games. However, their approach represents each case as an encoding of a single agent's perception and resulting action at a given time. Thus, they transfer the behavior of an agent when it perceives a situation similar to that of the observed agent. While these approaches do address knowledge transfer for robotic agents, they primarily operate at the strategic level. The goal of our work is to enable transfer at a lower level of control, where we transfer the demonstrated trajectory used to achieve a task.

Process-oriented CBR (PO-CBR) describes a similar approach, except that source cases represent a *set of actions* that comprise a task or workflow. CBR and PO-CBR differ in that PO-CBR operates over cases which represent *sequential* knowledge, and it is this series of actions that is adapted to address a new problem [59]. Not only does the content of each case need to be retrieved and adapted, but the internal structure of the sequence of events in the case must be preserved [60]. This approach has been applied to domains involving workflow planning, such as cloud management [61] and retrieving cases for business processes [62]. A primary issue in PO-CBR is that of retrieval; a retrieval method must be able to determine the similarity between the structure of two processes' actions, rather than merely the content or features of two cases.

2.3.3 Summary: Limitations of Existing Work in Analogical and Case-based Reasoning

Case-based reasoning relates closely to the problem of imitation learning, and provides a framework for modeling imitation from demonstration to adaptation and execution of a task in a new environment. PO-CBR also bears resemblance to the imitation learning problem in that case representation is centered around the sequence of actions (as in a series

of actions comprising a task demonstration). While analogical reasoning and CBR provide useful frameworks for selecting and adapting source cases for transfer based on their task differences, they are best suited for domains in which the action and state space representations are *already* abstracted such that the relationship between the source and target tasks is apparent. We next consider prior works that address this challenge of operating over low-level action and perception data.

2.4 Grounding Task Representations in Robot Actions and Perception

Transfer methods that operate over limited data, such as analogical reasoning, often rely on an abstracted representation of the task or goal. This produces an additional challenge when applied to the context of a robot's high-dimensional representations of action and perception. In this section, we consider prior work that addresses this challenge of *grounding* high-level representations of the robot's task model or state space in the robot's low-level perception of its environment.

2.4.1 Grounding Task and Action Models

Grounding a generalized task representation (e.g. "task recipes" [63, 64]) enables a robot to execute a high-level task representation in a specific environment. Bullard et al. [65] describe a method for grounding objects and semantic locations in perception using demonstrations. Tenorth et al. [66] ground task recipes in perception by identifying the object that best matches the features prototypical for that task step's object designator. Kulick et al. [67] present an active learning approach to train classifiers for task symbol grounding. While these approaches are useful for applying a task recipe to a specific environment, they rely on the robot having an abstracted task representation in the first place. Task recipes are designed to be generalizable across robots and environments; while a human with knowledge of the task can specify a task recipe by hand, a robot is unable to do so directly from task demonstrations.

2.4.2 Mapping Between Source and Target Environments

An alternative approach may focus on directly learning a mapping between the robot's source and target tasks and/or environments. Taylor et al. [32] propose that the robot may explore its state space in the target domain in order to derive an inter-task mapping between source and target state variables, using it to transfer its learned policy. While effective, this approach requires significant additional experience in the target domain, which we aim to minimize. Object-oriented Markov Decision Processes expressly encode the relation between objects in the robot's environment as a part of its state space representation [68]; however, this approach assumes that objects within the same classification play the same role within the task. Chernova and Veloso [69, 70] introduce confidence-based autonomy for policy reuse, in which the robot assesses its confidence in applying its current policy to a new state; if its confidence is below a threshold, it requests additional demonstrations from the teacher, otherwise acting autonomously from its current policy.

For task models that are defined with respect to specific objects in the robot's environment, these object references must be updated when one or more objects are replaced in the robot's environment. This requires a similarity metric that may be used to determine a mapping between objects in the source and target environments. Lee et al. [71] address mapping objects based on the similarity between the objects' point clouds. Their method infers a warping function that transforms the source object point cloud such that it closely matches that of the target object. However, using this warping function as a measure of similarity limits object mapping to those with similar shape. Huang et al. [72] address a similar problem of aligning object point clouds, but prioritize alignment of certain labeled features of the objects. These approaches are effective in identifying similar objects, but assume that (i) similar objects play the same role in the source and target environments and (ii) corresponding objects are comprised of the same parts or shapes. These assumptions do not hold when the agent does not know *a priori* which features to use in object mapping.

Some object replacements may have additional effects on task transfer, such as when

replacing an object that is used as a *tool* to complete the task. The shape of a tool alters its effect on its environment [41], and thus a tool replacement may necessitate a change in the manipulation of that tool in order to achieve the same task goal [73]. For tasks involving the use of a rigid tool, the static relationship between the robot's hand and the tooltip is sufficient for controlling the tool to complete a task [74, 75]. These methods assume a single tooltip for each tool, and that this tooltip is detected via visual or tactile means. For tasks involving multiple surfaces of the tool, the task model can be explicitly defined with respect to those segments of the tool, and repeated with tools consisting of similar segments [76]. However, this assumes a hand-defined model that represents the task with respect to pre-defined object segments, and that these object segments are shared across tools. Given enough training examples of a task, a robot can learn a success classifier that can later be used to self-supervise learning task-oriented tool grasps and manipulation policies for unseen tools [42].

2.4.3 Summary: Limitations of Existing Work on Task Grounding and Mapping

Task *grounding* is an alternative to task generalization, in which the objective is to execute a known task model in a new, specific environment. This often necessitates a mapping between the objects referenced in the task model and their counterparts in the target environment. The robot may infer the object mapping based on similarity between the objects' features, but this relies on hand-coded heuristics for object similarity that may not generalize across tasks. We similarly aim to situate a task model in the context of a new environment, but also aim to eliminate the assumptions that (i) the new objects are within the scope of the training examples (which would exclude atypical object replacements) and (ii) that the object features relevant to the task are observable and recorded by the robot.

2.5 Relationship to Thesis Contributions

We summarize the related work discussed in this chapter as follows:

1. **Learning from Demonstration**

   Benefits: Provides a method for quickly acquiring training data within the context of a new task and/or environment.

   Limitations: Not a scalable means of addressing *all* task or environment variations the robot may encounter.

2. **Transfer Learning and Few-Shot Learning**

   Benefits: Addresses task and/or environment variations by directly learning the relationship between task changes and their effects on task execution.

   Limitations: Typically requires extensive training datasets or a simulator in which the agent can generate additional training data, neither of which are readily available for a newly-learned task.

3. **Analogical and Case-Based Reasoning**

   Benefits: Models the relationship between source and target problems, and is well-suited for data-sparse domains.

   Limitations: Relies on the task and action models being represented at an abstracted level (e.g. in terms of goals or preconditions) that is not explicitly conveyed through demonstrations.

4. **Task Grounding and Mapping**

   Benefits: Rather than learn a fully-generalizable task model, these approaches ground an existing task model in a *specific* new environment, and thus does not need to be re-trained in the target environment.

   Limitations: Relies on hand-coded assumptions about the effect of environment changes on task execution.

The remainder of this dissertation focuses on learning a mapping between the robot's source and target environments *without* relying on hand-coded mapping heuristics. To do so, we aim to leverage the grounded, contextual knowledge imparted by demonstrations,

while also maximizing the robot's autonomy by limiting the amount of interactive assistance required by the robot. This requires an understanding of (i) the data that should be contained in the mapping and (ii) the modes of interaction that the robot may use to obtain this data. In the next chapter, we analyze the relationship between these two factors, as well as how they are dependent on the similarity between source and target environments.

CHAPTER 3

TAXONOMY AND REPRESENTATION OF TRANSFER PROBLEMS

A robot that operates in human environments will encounter an unbounded number of environment variations, ranging from variations in the objects and perceptual features in its environment, to object replacements that introduce new task constraints. In Chapter 2, we summarized existing approaches to generalizing a robot's task models to a new environment. Many of these approaches either (i) require extensive training datasets or (ii) make assumptions about the relationships between environment changes and their effect on task execution. Rather than rely on these assumptions, we aim to leverage the grounded, contextual knowledge imparted by demonstrations, while also maximizing the robot's autonomy by minimizing the amount of assistance required by the robot. This first requires an understanding of the data that must be obtained by the robot in order to transfer its task model to a new environment. Furthermore, how the robot requests information from the teacher affects the data it obtains from the teacher's assistance. In this chapter, we analyze how the similarity between the source and target environments effects both these attributes (data needed to enable transfer, and the interaction method used to obtain that data).

This dissertation focuses on the effect of changes in the robot's *environment* on task execution, and does not address transferring other aspects of the task, such as reward functions or task goals. Within the scope of addressing environment changes, there are several sources of novelty that are introduced by object changes or replacements. For example, changes in the location, dimensions, 3D shape, and/or affordances of a new object must all be addressed for the robot to transfer a task model successfully to the new environment. However, these changes will affect the task in different ways. For example, relocating an object will have a minor effect on task execution compared to replacing one tool object with another (in which case, the task adaptation depends on how the tool is used within the

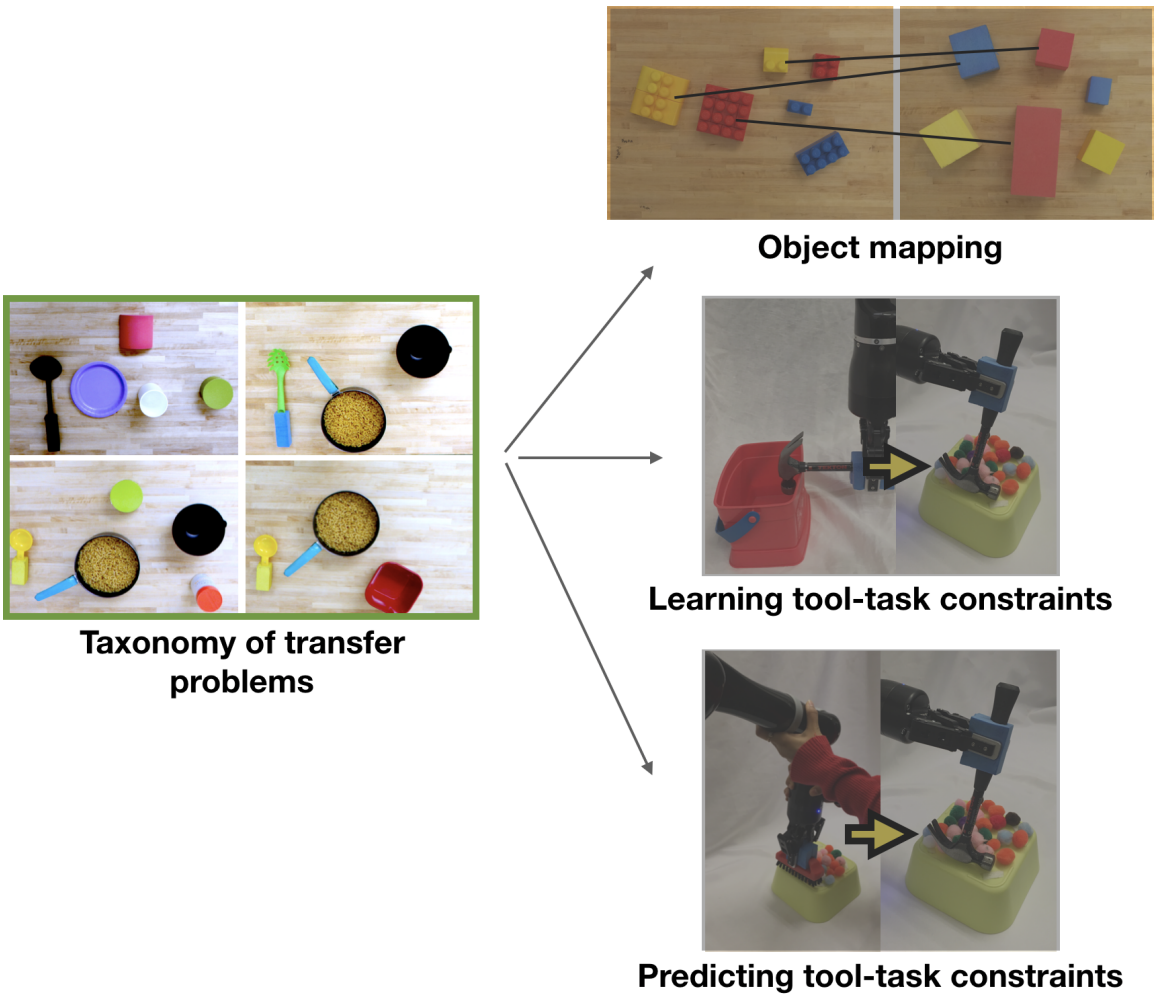


Figure 3.1: We first present a taxonomy of transfer problems that highlights the relationship between environment similarity, task abstraction, and the data required to ground the abstracted representation in a new environment. This taxonomy (highlighted in green) motivates later work on three specific categories of transfer problems (shown at right).

context of the task). As a result, the type of novelty encountered in a target environment affects how a robot should address this novelty.

A human teacher can provide demonstration samples of these environment variations a priori. However, as discussed in Chapter 2, these demonstrations are not sufficient to span the full space of environment variations the robot may encounter, nor their effect on task completion. We consider a data-sparse paradigm instead, where the robot transfers a *specific* task model from a source environment in order to address a related target environment containing a new set of objects.

In this chapter, we present a taxonomy of transfer problems based on the similarity between source and target environments. We propose a task representation that supports abstraction in order to address three categories of task transfer problems. Figure 3.1 illustrates the relationship between this chapter and the remainder of the thesis. In later chapters, we demonstrate how the abstracted representation can be grounded via interaction.

3.1 Categorizing Task Differences

We refer to a *task* as a sequence of object-oriented task steps or *skills*, each consisting of their own action model and performed in series to achieve a goal. As an example, a *cup-pouring* task would consist of three action models: (i) grasping the cup, (ii) lifting the cup, and (iii) tipping the cup, each defined with respect to the cup's pose in the robot's environment. This definition results in three key elements of a task representation: the robot's state with respect to its environment (e.g. objects), the action model comprising each skill, and the goal that is achieved by its execution. These correspond to the state space, action space, and goal/rewards commonly used to define a Markov Decision Process or other task-planning problem. We next consider how changes to any of these three defining task elements affect the robot's execution of the task.

3.1.1 Goal Space Changes

A representation of the task goal(s) may be used to guide transfer by defining the preconditions, postconditions, or constraints that must be met to successfully complete the task. Analogical reasoning is a cognitively-inspired approach that is well suited to adapt to changes in the goal representation. Prior work in this research area [47, 48, 49] operates over a set of *source cases*, each containing an instance of a problem-solution pair, stored in a *source case memory*. An unfamiliar problem is then addressed using the following methodology. As new problems are viewed, the single most similar observed case is pulled from the source case memory. The retrieved case is then compared to the new, *target case*,

and a mapping is derived that contains the differences between the two. Using this mapping, the retrieved source case solution is then adapted to address the differences between the two cases. The adapted solution is then deployed in the context of the target case.

A central goal of analogical reasoning is identifying the common relationships between problem-solution pairs. This relies on having a representation of both the problem and solution that is abstract enough for these relationships to be identified. A task goal that is defined symbolically (e.g. as a set of pre-/post-conditions) could support analogical reasoning; however, this requires that the symbolic representation be learned or otherwise defined (e.g. by a human teacher or a dataset).

Prior work in Learning from Demonstration has shown how a robot may learn a goal model through demonstrations, resulting in a representation such as a probabilistic goal model [77], a series of task constraints [78], a goal descriptor [79], or an abstracted skill tree [80]. These methods require multiple demonstrations of various successful and unsuccessful goal states, and/or goal specifications by a human teacher. With limited demonstrations, however, the goal representation is not immediately learnable. Unless this goal is manually specified by a human, we presume that the robot does not have access to a goal model to facilitate transfer of its learned action models to a target problem.

3.1.2 Action Space Changes

Action space changes occur when transferring a learned task model to a robot with different kinematics, constraints, and/or output modalities. Additionally, new kinematic constraints may be introduced or removed, also resulting in a change in the robot's action space. Transfer learning methods have been used to address these task differences with the goal of transferring knowledge gained in the source domain to improve an agent's performance and/or learning speed in the target domain [31, 32]. In relation to skill learning, this may involve transferring an action policy that is learned in one domain such that it can be used to reduce the time required to repeat or relearn the skill in a second domain, such as

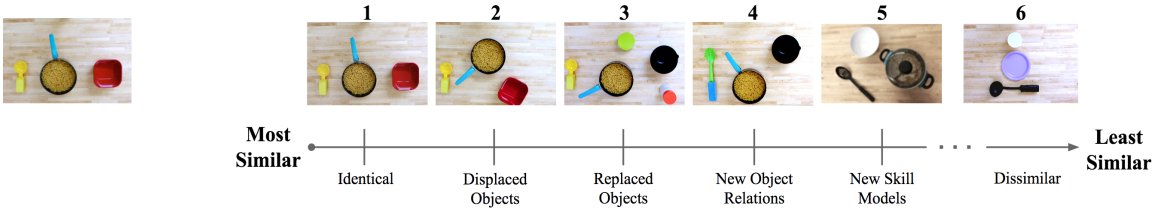
in [81]. Transfer via inter-task mapping [32, 82] enables transfer learning for reinforcement learning agents with similar goals but different action spaces in domains such as RoboCup Keepaway. Within the context of a single robot that operates in human environments, we do not expect that the robot's action space will change, and thus do not address this category of transfer problems.

3.1.3 State Space Changes

In this dissertation, we primarily address transfer over changes in the robot's state space. The robot's state space is typically defined in terms of the relationship between the robot's kinematic state and its environment. Regardless of the exact state space specification used, the environmental variations that are common in human environments are likely to be reflected in the robot's state space as well. We categorize these variations as follows:

- *Structural* changes in which the relationship between objects within the robot's environment is altered. E.g. when objects are moved around the scene.
- *Perceptual* changes in which the robot's environment appears different while remaining structurally and/or semantically the same. E.g. changes in lighting or in the appearance of an object.
- *Semantic* changes that affect the relationship between the robot and its environment. E.g. the introduction of obstacles that constrain the robot's motion, or the use of an object as a new end-effector.

Prior work has addressed the problem of structural changes in the robot's state space. Pastor et al. [27] describe an approach to learning a series of primitive skill models which comprise complex tasks. A Dynamic Movement Primitive (DMP) is trained over a demonstration by perturbing a linear spring-damper system according to the velocity and acceleration of the robot's end-effector at each time step [25, 27]. By integrating over the DMP, a trajectory can then be generated that begins at the end-effector's initial position and ends at



(a) Source (b) Potential Target Environments

Figure 3.2: Spectrum of Similarity Between Source and Target Environments

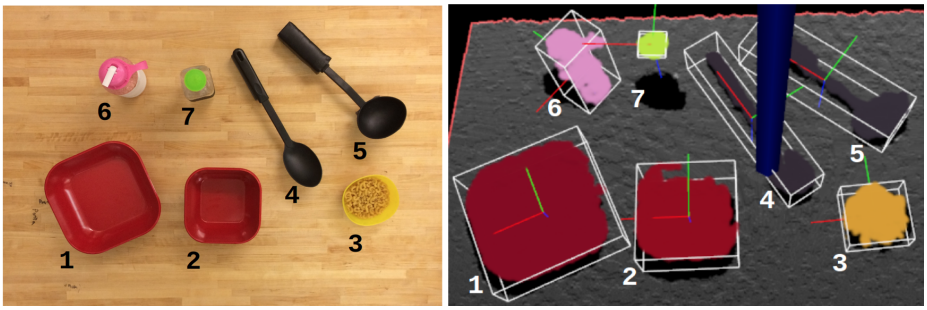


Figure 3.3: An overhead view of a table-top environment (left) and the segmented point cloud representation (right)

a specified end point location. Thus, after training a DMP, the only parameter required to execute the skill is the desired end point location. By parameterizing the end point location of each DMP skill model according to object locations, the overall task can be generalized to accommodate new object configurations.

3.1.4 Relationship Between Abstraction and Similarity

Figure 3.2b illustrates how these state space changes may be expressed. A task demonstrated in a source environment (e.g. a scooping task performed in the environment shown in Figure 3.2a) can be directly reused in a target environment which either (i) does not require modification of the learned task (image 1 in Fig. 3.2b), or (ii) has a known parameterization according to the target environment. For example, image 2 in Figure 3.2b demonstrates a target environment containing a *structural* change: repositioned objects. If the robot has learned a task model that is parameterized with respect to the location of objects in the scene, it can address this target environment by re-evaluating its parameter

values according to the objects' new positions.

When addressing a target environment that exhibits *perceptual* changes (image 3 in Fig. 3.2b), the objects in that environment are unfamiliar but serve the same purpose as objects in the source environment and do not need to be manipulated differently. This problem can be addressed by identifying a mapping between objects in the source and target environments. This mapping can be used to ground the task parameters in the target environment's feature values rather than the source environment.

Image 4 in Figure 3.2b differs from the source (Figure 3.2a) in that objects are: (i) displaced, (ii) replaced, and now (iii) *constrained.* This constraint is a result of the role that the scoop object plays in the task; the scoop is used as an end-effector during a scooping task, and thus changing the size of the scoop affects how the robot should complete that task. The robot's actions must now be constrained such that its end-effector remains higher above the table in order to complete the task successfully. This presents a *semantic* change, since the relationship between the robot and its environment has been changed due to the tool replacement. Since the task model was trained with respect to the source environment (and as a result, the source scoop tool), its output must be mapped to the corresponding actions using the target scoop.

Image 5 in Figure 3.2b differs from the source in similar respects, with one additional difference: an extra step is needed in order to lift the lid off the pasta pot prior to scooping the pasta. As a result, the original skill models learned in the source cannot be directly transferred. In addition to deriving an object mapping and action mapping as in the previous transfer problems, this target environment also requires that the robot derive or learn a new action model to account for the missing step.

These task differences illustrate a *spectrum* of similarity between the source and target; at one end of the spectrum, the source and target differ in small aspects such as object configurations. At the other end of the spectrum, they contain more differences, until finally (as in image 6 in Figure 3.2b), the target environment cannot be addressed via transfer.

While we have highlighted discrete levels of similarity in this spectrum, we do not claim this to be an exhaustive categorization of transfer problems. In prior work, we have also explored how a robot may need to exhibit creativity in order to address more dissimilar target environments [83]. In summary, Figure 3.2 illustrates that without further information about the task as it pertains to the target environments, task transfer methods are limited to addressing a narrow set of transfer problems: those in which the target environment does not require novel behavior or reasoning to address.

3.2 Approach: Tiered Task Abstraction

The previous section described how task differences affect the robot's task completion differently, and thus require different information in order to transfer the learned task model to a new environment. We propose the Tiered Task Abstraction (TTA) task representation to address this range of transfer problems. We aim for this representation to reflect the relationship between (1) changes in the state space and (2) their effect on the task transfer.

The TTA representation uses a demonstration trajectory as input, recorded as a discretized series of poses. This trajectory may be a dense series of poses if recorded as a continuous motion, or sparse if recorded as distinct keyframes indicated by the teacher throughout the demonstration. This trajectory may be modeled as a single action model according to a set of basis functions, or alternatively, as a set of different action models *each* trained according to single target pose. Assuming the task is object-centric, we model each target pose or basis function center with respect to one or more object poses. We refer to this relationship as the model's *parameterization function*, defined as a linear function over a set of object features. The identity of these features may differ between task executions, due to perceptual variations (e.g. lighting or orientation changes) or object replacements. Rather than rely on persistent object and feature labels, we incorporate a *feature selector* function that returns the labeled, relevant features to be used in the parameterization function.

	 Non-Abstracted	Abstraction 1	Abstraction 2	Abstraction 3	Dissimilar
Retained Knowledge	Action Models Parameterization Functions Feature Selectors Feature Values	Action Models Parameterization Functions Feature Selectors	Action Models Parameterization Functions	Action Models	None
Grounded Knowledge	None	Feature Values	Feature Selectors Feature Values	Parameterization Functions Feature Selectors Feature Values	Action Models Parameterization Functions Feature Selectors Feature Values

Figure 3.4: The grounding requirements for each level of abstraction

Overall, the Tiered Task Abstraction representation consists of four elements: an action model, parameterization function, feature selector, and feature values. These four elements represent a task demonstration as a single action model as follows:

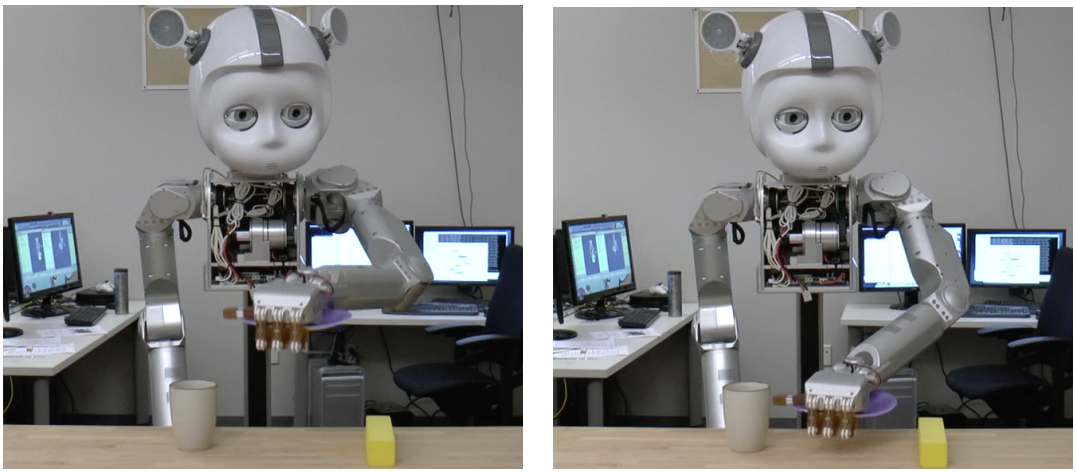
$$a_i(p_0(f_0(E)), p_1(f_1(E)), \ldots p_m(f_m(E))$$

Or as a *series* of action models as follows:

$$a_0(p_0(f_0(E))), a_1(p_1(f_1(E))), \ldots, a_m(p_m(f_m(E)))$$

where a_i is a single action model based on the parameterization function p_i which operates over the features returned by f_i from the full set of feature values E.

Note that each element is parameterized by the next; by omitting one or more elements from the task representation, the resulting representation is one that is *abstracted*. In doing so, a task can be represented at a level of abstraction which is common to both the source and target environments. Figure 3.4 defines three abstractions of this representation. However, once a representation is abstracted, it must be *grounded* in the target environ-



(a) After Task Step 1 (b) After Task Step 2

Figure 3.5: Steps Comprising the *Table-Setting* Task

ment in order to produce an output that is executable by the robot. In an embodied system, grounding refers to parameterizing a representation based on perception in the physical world. A representation is *grounded* in a target environment when all of its elements (action model, parameterization function, feature selector, and feature values) are present and defined based on information derived in the target environment (either by perception or interaction in the target environment). We summarize the grounding requirements of each abstraction level in Figure 3.4.

3.3 Evaluation: Transferring a Task Model at Multiple Abstraction Levels

We evaluate whether the Tiered Task Abstraction reflects the relationship between task differences and the resulting data requirements to enable task transfer. To do this, we represent the same task model at three abstraction level and test its performance over three variations of that task. We test performance on two tasks: a *Table-Setting* task and a *Scooping* task.

3.3.1 Experimental Setup

We evaluated our approach on the Curi robot shown in Figure 3.5. Curi is equipped with two arms consisting of 7 degrees-of-freedom and an under-actuated gripper. We used only

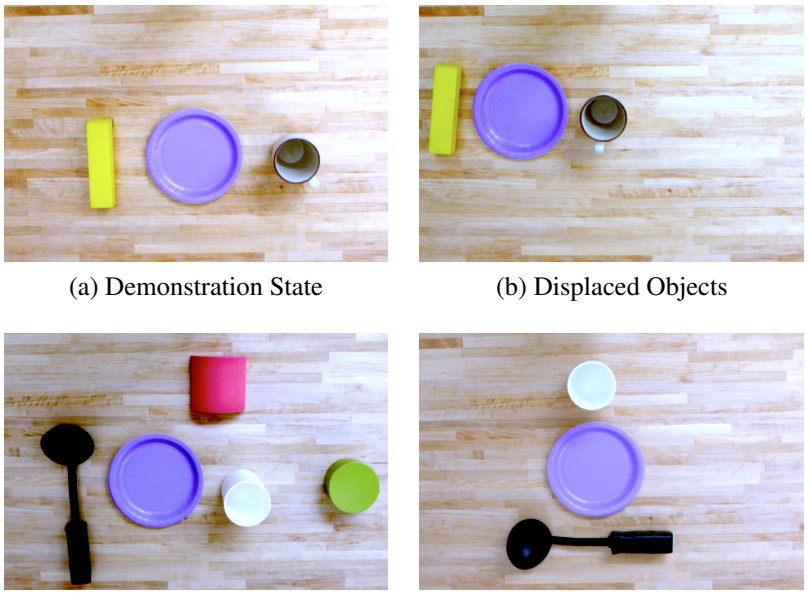
the robot's left arm for demonstrations and task execution. During demonstrations, we used a gravity-compensating controller so that the robot's arm could be easily moved to complete the task. The robot perceived its tabletop workspace using an overhead RGBD camera (not pictured) which provided a top-down view of the table.

For this experiment, we defined each element of the TTA representation as follows:

- Action Models: We demonstrated each task on a single, 7 DOF arm on the robot shown in Figure 3.5. Each demonstration was recorded as a single, continuous trajectory which was then segmented manually into several task steps. We trained a task model over each step as a Dynamic Movement Primitive (DMP), which can be re-parameterized for a new task configuration by specifying the new start and end poses for the desired trajectory.

- Parameterization Functions: We defined the parameterization function for each task model as the end-effector's position with respect to the nearest object in the robot's environment. This reflects the constraints guiding the end-effector's start and end position at each step of the task as an offset from an object location. For example, suppose that one segment of a scooping task ends with the robot's end-effector positioned 5 cm above the pasta bowl before continuing with the next task step. The corresponding parameterization function is: $<o_x, o_y, o_z + 5>$, where o is a reference to the relevant object (in this case, the location of the pasta bowl). The robot recorded these object poses (and subsequently, the parameterization functions with respect to those object poses) autonomously using an RGBD camera located above its tabletop workspace. When transferring the TTA representation at an abstraction where the parameterization function must be grounded in the target environment, we manually re-define this 3D offset.

- Feature Selectors: The robot assigns a unique, non-descriptive object ID to the segmented objects in its environment. These object IDs are referenced in the aforemen-



(a) Demonstration State (b) Displaced Objects

(c) Replaced Objects (d) Rotated Scene

Figure 3.6: Variants of the *Table-Setting* Task Environment

tioned parameterization functions. When transferring the TTA representation at an abstraction where the feature selectors must be grounded in the target environment, we manually provide a one-to-one mapping between object IDs in the source and target environments.

- Feature Values: These are the feature values associated with each object label. We define these as the bounding box dimensions and position of each object in the robot's environment. We use the algorithm described in [84] to segment the RGBD point-cloud into a set of bounding boxes surrounding each object above the tabletop surface plan. When transferring the TTA representation at an abstraction where the feature values must be grounded in the target environment, the robot autonomously updates the feature values from its RGBD sensor input.

3.3.2 Table-Setting Task

In the first experiment, the robot learned a table-setting task in the environment shown in Figure 3.6a. The table-setting task consisted of placing a plate between a cup and utensil (represented by the yellow block), requiring two skill models encoding (i) moving the end effector to a point between the cup and utensil (Figure 3.5a), and (ii) setting the plate down (Figure 3.5b).

Training

The training portion of the experiment was run as follows:

1. At the start of the demonstration, the location of each object was recorded via an overhead RGBD sensor.

2. Throughout the demonstration, the robot recorded its end-effector pose relative to the robot's base pose.

3. We manually segmented the demonstration using voice commands, resulting in two separate trajectories: one to position the plate, and another to lower it onto the table.

4. At the end of each task segment, the robot recorded the transform position between the end-effector and the object closest to it.

5. Following the demonstration, we trained a DMP over each of the two trajectories.

Testing

We evaluated task performance on three transfer categories:

1. *Displaced-Object environments*: Contain the same objects as in the original demonstration, but displaced as shown in Figure 3.6b.

2. *Replaced-Object environments*: Contain cup and utensil objects that are different than those used in the original demonstration. Additional "distractor" objects are also provided that are irrelevant to completing the skill. An example is shown in Figure 3.6c.

3. *Rotated-Scene environments*: Contain the cup and utensil objects as in the replaced-object environment, but with the cup and utensil jointly rotated 45-90 degrees away from the robot. An example of this is shown in Figure 3.6d. This has the effect of requiring that the robot fulfill the object relation of placing the plate between the two other objects, rather than simply placing the plate to the left of the cup as in the previous target environments.

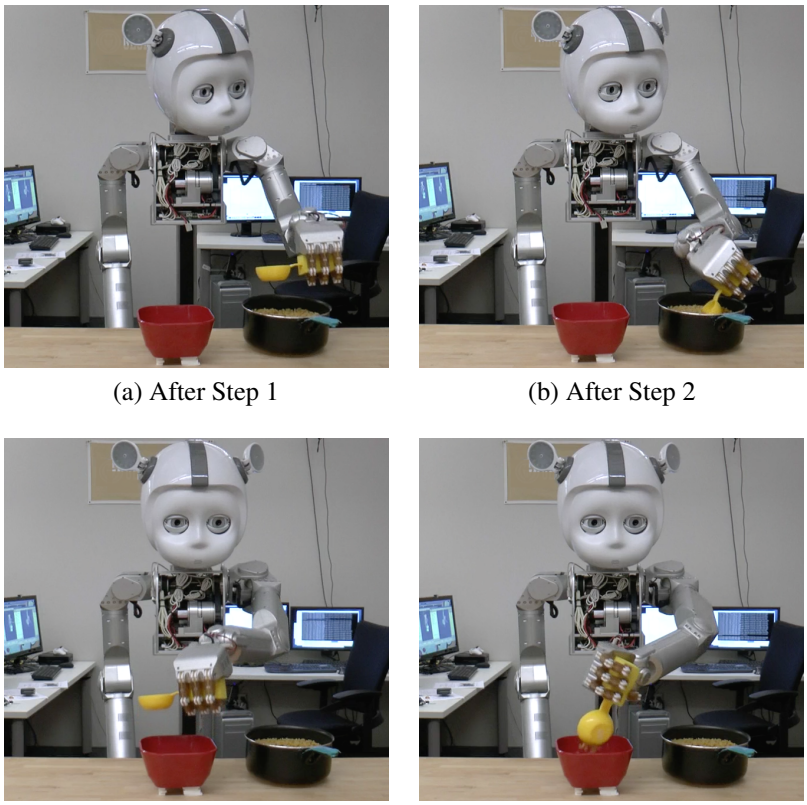
These categories of target environments correspond to the feature sets listed in Figure 3.4. We represented the task model at *three* levels of abstraction, and evaluated each abstraction on *ten* target environment variations in each of the *three* environment categories, resulting in a total of *90* transfer evaluations for the table-setting task. A "success" was noted each time the plate was placed between the cup and utensil without the plate touching either object.

3.3.3 Scooping Task

The second experiment revisited the scooping task environment depicted in Figure 3.8a. The scoop task consisted of four skills: moving the scoop from the initial position at the robot's side to the pasta bowl (Figure 3.7a), scooping the pasta (Figure 3.7b), moving the scoop to the target bowl (Figure 3.7c), and then pouring the scoop over the target bowl (Figure 3.7d).

Training

We performed the training portion of this task similar to that of the table-setting task, but with the robot grasping the yellow scoop prior to starting the demonstration. Since the



(a) After Step 1 (b) After Step 2

(c) After Step 3 (d) After Step 4

Figure 3.7: Steps Comprising the *Scooping* Task

scooping task is more complex than the first task, we recorded three demonstrations, keeping the demonstration that yielded the most stable performance when re-tested in the source demonstration environment.

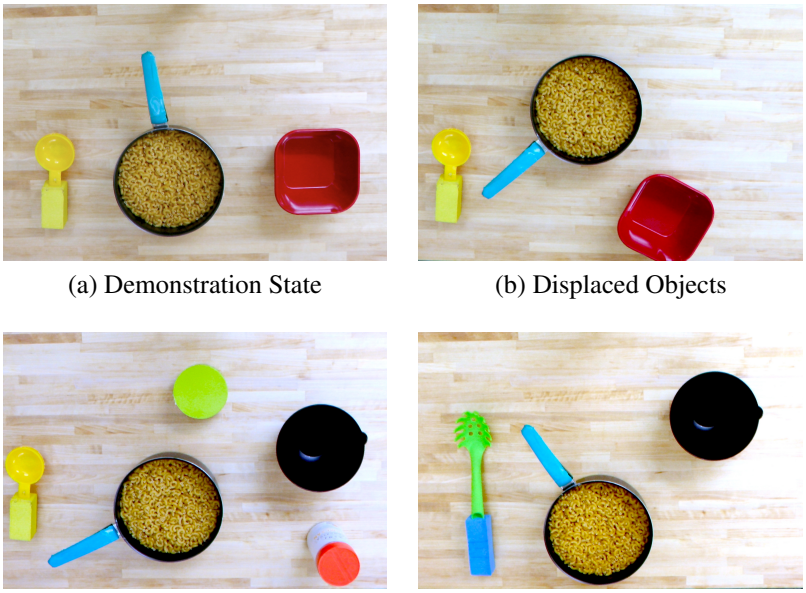
Testing

We evaluated the robot's performance in three transfer categories:

1. *Displaced-Object environments*: Contain the same objects as in the original demonstration, but displaced as shown in Figure 3.8b.

2. *Replaced-Object environments*: Contain a different target bowl and a set of additional, "distractor" objects that are irrelevant to completing the task, as in Figure 3.8c.

3. *Replaced-Scoop environments*: Contain the same target bowl as in the replaced-



(a) Demonstration State (b) Displaced Objects

(c) Replaced Objects (d) Replaced Scoop

Figure 3.8: Variants of the *Scooping* Task Environment

object environment, but also contain one of two scoops with longer handles than the one used in the original demonstration, as in Figure 3.8d.

The *three* task abstractions were each applied to transfer the task *ten* times per each of the *three* environment categories, resulting in a total of *90* transfer evaluations for the scooping task. A "success" was noted each time any amount of pasta was moved to the target bowl without the target bowl being tipped over.

3.4 Results: Applying the Tiered Task Abstraction to Task Variations

Figure 3.4 summarizes the three abstraction levels evaluated in this experiment. For both the table-setting and scooping tasks, we hypothesized that (i) Abstraction 1 could only consistently address the displaced-objects environment, (ii) Abstraction 2 could consistently address the displaced and replaced objects environments, and (iii) Abstraction 3 could consistently address all three environments.

Table 3.1: Success Rates for Each Approach Applied to the Table-Setting Task

	Retargeting	Mapping	Relational
Displaced Objects	10/10	9/10	10/10
Replaced Objects	0/10	10/10	10/10
Rotated Scene	1/10	4/10	7/10

3.4.1 Table-Setting Task

Table 3.1 provides the success rate of each transfer method when applied to 10 varying instances of each category of table-setting environments. As expected, Abstraction 1 succeeded consistently on *only* the displaced-objects environment. Abstraction 2 resulted in consistent performance in the first two environments, and additionally, succeeded in a few of the rotated-scene scenarios. Transfer at this abstraction level succeeded in the few occasions when the robot was able to place the plate to the left of the cup without the plate touching either the cup or utensil. This demonstrates that while this abstraction level may be used to address some of the rotated-scene environments, it cannot do so consistently.

In the three scenarios in which Abstraction 3 did not succeed in addressing a rotated-scene environment, the front of the robot's hand had hit the cup, leaving the plate close to the intended location but not quite meeting the threshold for successful task completion. We anticipate that this abstraction could be used more successfully if the parameterization function used to ground this abstraction had incorporated information about the size of the cup and the robot's hand to avoid hitting other objects.

3.4.2 Scooping Task

Table 3.2 provides the success rate of each abstraction level when applied to target environments in three categories of scooping task problems. As in the previous results, the displaced-objects environments could be addressed consistently using any of the three ab-

Table 3.2: Success Rates for Each Approach Applied to the Scooping Task

	Retargeting	Mapping	Relational
Displaced Objects	10/10	10/10	10/10
Replaced Objects	0/10	10/10	9/10
Replaced Scoop	0/10	0/10	8/10

Table 3.3: Success Rate Comparison Across Abstractions, Tasks, and Transfer Categories

	Table-Setting Task			**Scooping Task**		
	Disp. Objects	Replaced Objects	Rotated Scene	Disp. Objects	Replaced Objects	Replaced Scoop
Abs. 1	**10/10**	**0/10**	**1/10**	**10/10**	**0/10**	**0/10**
Abs. 2	**9/10**	**10/10**	**4/10**	**10/10**	**10/10**	**0/10**
Abs. 3	**10/10**	**10/10**	**7/10**	**10/10**	**9/10**	**8/10**

straction levels, and the replaced-objects environment could only be addressed consistently by Abstractions 1 and 2. Finally, these results also indicate that Abstraction 3 succeeded consistently across all three classes of transfer problems.

3.5 Tradeoff Between Generality and Data-Efficiency

These results suggest that Abstraction 3 provided the most consistently successful results across the full range of transfer problems we tested, with Abstractions 2 and 1 each addressing fewer transfer problems, respectively. However, the more that the task is abstracted, the more data is required to ground that abstraction in the target environment. Abstraction 3 requires both an object mapping and parameterization function in order to ground this abstraction in the target environment. While this grounded data was provided manually in this experiment, we intend for the robot to eventually learn this data either autonomously or from more indirect assistance (such as a human teacher providing additional task demonstrations or answering the robot's questions). These results indicate a **trade-off between the generality of a task representation, and the amount of additional information required to ground that task representation in the target environment**. Furthermore,

these results support our hypothesis that there is a **correlation between (i) the level of similarity between the source and target environments and (ii) the level of abstraction that should be used to address a transfer problem**.

We also note that abstraction occurs within the DMP action model itself. DMPs are intended to control the robot's end-effector position, which is itself an abstraction of the robot's motion in joint-space. Furthermore, DMPs model an end-effector trajectory as a point-attractor system that is perturbed by the centering and weighting of several basis functions that are temporally activated throughout the trajectory. As a result, the point-attractor system enables the trajectory to be guided by a start and end position that may be modified, while also maintaining the general shape of the trajectory. This enables an abstraction of the goal parameterization that is separate from the dynamics parameters, making it ideal for use in the TTA representation. While another action model may be used, it may need to be accompanied by an additional parameterization function in order to enable this separation of goal parameters from dynamics parameters.

3.5.1 Grounding Task Abstractions

Figure 3.4 summarizes the representation elements which must be retained or grounded for each category of transfer problems. Our experiment demonstrates the effectiveness of each abstraction level on a range of task variations, with each abstraction being grounded manually. For a robot that operates in human environments, we aim for the robot to be able to ground its own task abstractions using continued interaction with a human teacher during task transfer. This would enable the robot to leverage the human teacher's knowledge of the task domain in order to perform transfer. This relationship between (i) environment similarity and (ii) assistance from the human teacher introduces a second dimension to the aforementioned similarity spectrum; **as the source and target environments become more dissimilar, the robot's level of transfer autonomy decreases and its dependence on interaction with the human teacher increases**.

As discussed in the previous section, the first two categories of transfer problems (e.g. identical and displaced-objects environments) could be addressed by the robot with full autonomy. In order to address more difficult transfer problems, the robot must ground both the (i) parameterization functions and (ii) skill models in the target environment. These are the two elements of the TTA representation which contain the most high-level information about the task: the constraints between the robot's end-effector and objects in the environment, and the action model which preserves the trajectory shape of the demonstrated action, respectively. These represent high-level information about the task, and require knowledge of the task goal to define. As a result, we do not expect that this data can be grounded by the robot with complete autonomy, but rather, may be obtained using input from a human teacher.

3.6 Summary

In this chapter, we have:

1. Introduced transfer not as a single problem, but rather as a series of problems that range in difficulty according to the similarity between the source and target environments.

2. Defined a relationship between (i) the similarity between source and target environments, (ii) the effect of this similarity (or dissimilarity) on the robot's task execution, and (iii) the level of abstraction at which the task model should be represented in order to enable transfer to that target environment.

3. Defined the Tiered Task Abstraction representation: a task representation that, when abstracted, is capable of addressing transfer problems with varying dissimilarity between the source and target environments.

4. Presented experimental results that illustrate the effect of transferring a task model at several levels of abstraction, and the resulting performance over a set of transfer

problems ranging in their difficulty.

3.6.1 Key Contributions and Insights

This chapter is the first to analyze task transfer in the data-sparse context of robot learning from demonstration. We have evaluated transfer performance using three levels of abstraction, each of which was grounded manually for the target environment. This serves to answer the question of *whether* we can use a single task representation to address a range of transfer problems via abstraction.

Insight #1: The more dissimilar the source and target environments are, the more that the source task representation must be abstracted to be successfully transferred to the target environment.

Insight #2: There is an inverse correlation between (i) the degree to which the task representation is abstracted and (ii) the amount of data that is needed to ground the abstracted representation in the target environment.

Insight #3: As a result of #1 and #2, there is a tradeoff between the generality of a task representation (e.g. the range of transfer problems that it can successfully address) and the data requirements that must be met to ground the abstracted task representation in the target environment.

3.6.2 Open Questions

We now consider the question of *how* to apply an abstracted task representation to a specific transfer problem; particularly, how to ground that abstraction without requiring a human to manually define it for the robot. Presuming that the human teacher is aware of the goal of the task, and how that goal should be met in the target environment, we posit that the teacher is available to assist the robot in reaching that goal. It is advantageous for the robot to continue to interact with the human teacher in order to ground these representation elements, since the teacher does know how the task should be performed to achieve the

task goal. The aim of interactive grounding is to produce a solution that (i) is partially autonomous (the robot interacts with a human teacher and may receive additional instruction, but does not require a full re-demonstration of the task), (ii) enables collaboration with the human teacher so that the robot may infer information about the task in the target environment, (iii) results in parameterization functions and/or action models that can ground an abstracted task representation, and (iv) grounds the TTA representation such that a trajectory can be executed in the target environment.

We hypothesize that there are several interactive approaches to task grounding. For example, the robot may use speech as the assistance modality by asking about objects prior to attempting to perform the task. Alternatively, the robot could instead rely on the teacher to correct its actions (rather than proactively ask for assistance) after each task step. Transfer problems of increased difficulty may be also addressed via exploration, in which the robot collaborates with the human teacher to explore new actions, to which the human teacher can respond by guiding the robot's exploration or providing feedback.

In the remainder of this dissertation, we analyze the specific categories of transfer problems introduced in this chapter. For each category, we define (i) the level of abstraction used to address that category, (ii) an interaction method which the robot may use to obtain task-specific knowledge that grounds its abstracted representation, and (iii) an algorithm for filtering and modeling the information obtained from interaction such that it can be used to ground the abstracted representation.

CHAPTER 4

HUMAN-GUIDED OBJECT MAPPING FOR TASK TRANSFER

In the previous chapter, we presented a relationship between task similarity, task abstraction, and the data requirements for grounding that abstracted representation in a new environment. We now focus solely on the problem of *grounding* an abstracted task representation via interaction with a human teacher. Specifically, we address the category of transfer problems in which the robot requires a mapping between objects in the source and target environments in order to transfer its task model. Figure 4.1 illustrates the relationship between the focus of this chapter and the remainder of the thesis.

To succeed in target environments containing new objects, the robot must identify correspondences between the new objects and those in the original environment, a problem known as *object mapping*. This is a complex problem, however; given n objects in each environment, there are $n!$ possible mappings to consider, in principle. In this chapter, we focus on this object mapping problem. We assume that a task model was previously learned (by demonstration or otherwise), and do not address the problem of goal learning or the entire process of task transfer at this time. Rather, by evaluating object mapping independently of any specific task learning algorithm, we demonstrate that it would be compatible with a variety of learning algorithms, provided that they produce a list of object-directed steps for each task.

While object mapping has been addressed in prior work (see Chapter 2), it is typically assumed that the robot can gain additional experience in the target environment, or knows which object feature(s) to use to evaluate object correspondences. However, in the context of a robot that learns a task from human interaction, such assumptions lead to two challenges. First, gaining additional experience in the target environment can be a time-consuming task in which the human teacher must continue to provide task demonstrations

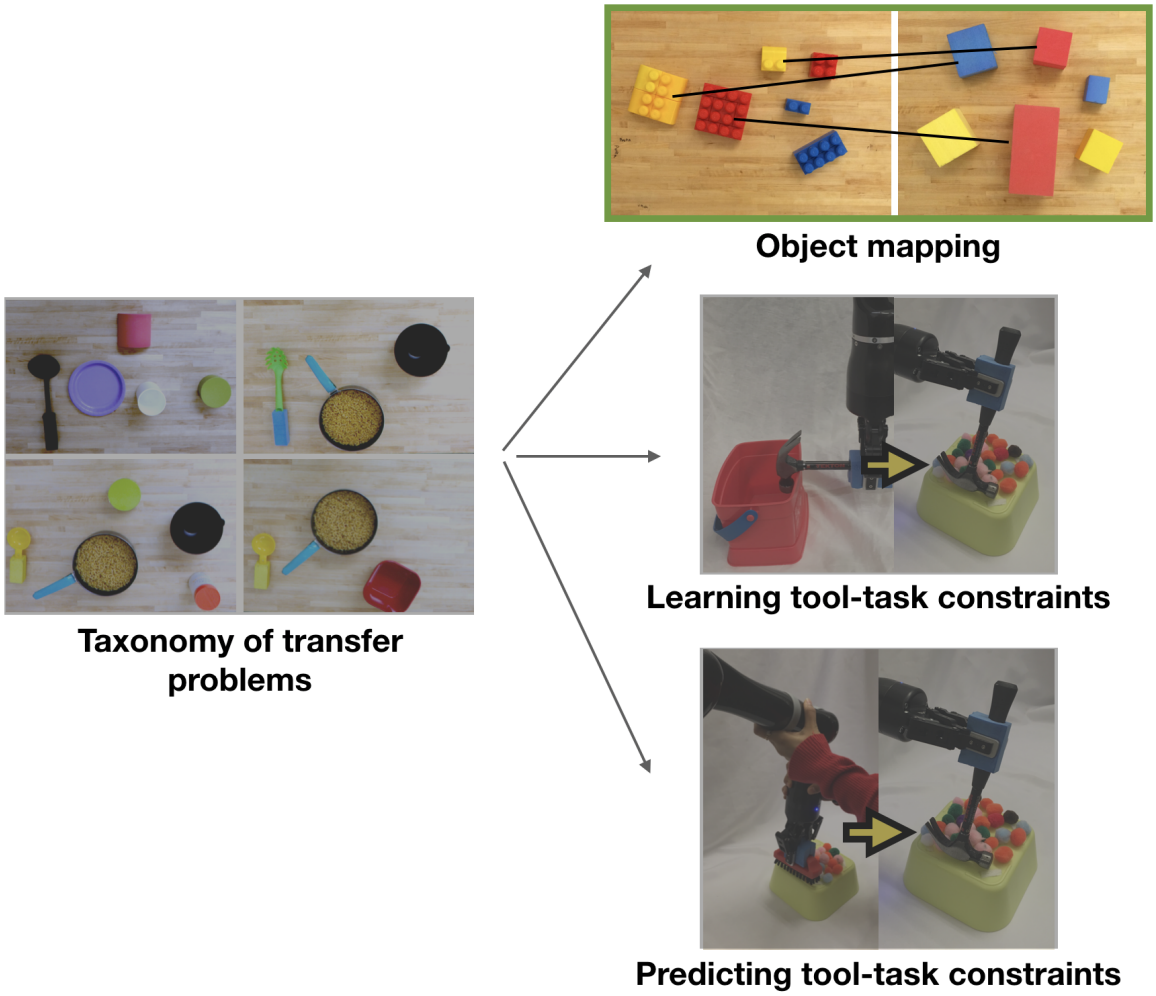


Figure 4.1: After identifying a taxonomy of transfer problems, we now focus on a specific category of transfer problems, highlighted in green: those requiring an object mapping in order to ground the abstracted task representation.

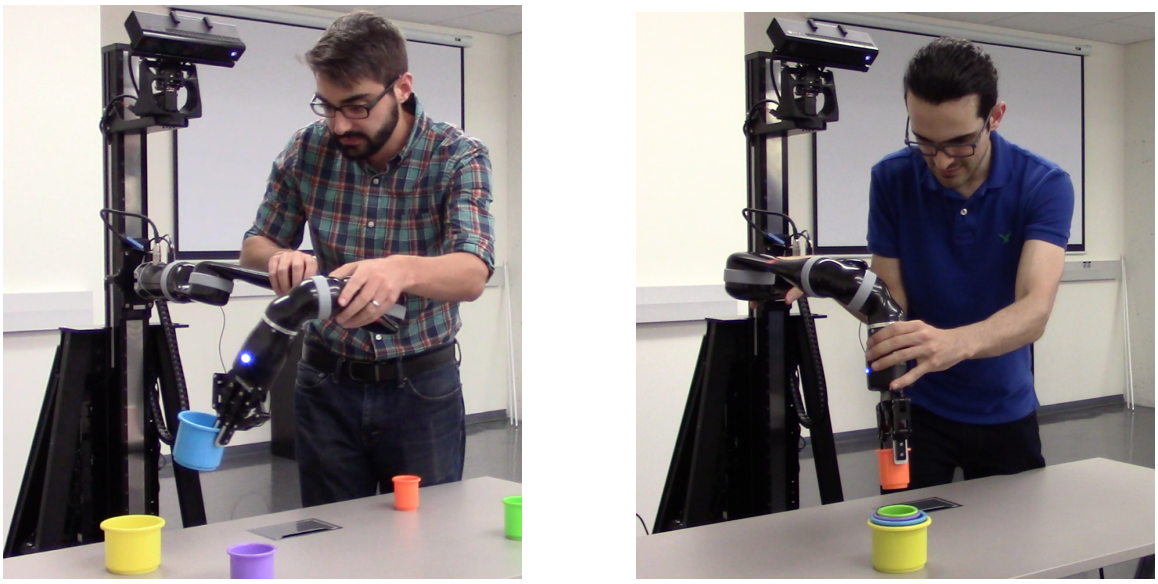


Figure 4.2: User study participants providing interactive task demonstrations to the robot

or feedback to the robot as it explores the environment. Simply helping the robot gain the correct mapping would more efficiently utilize the teacher's time. Second, the robot does not have the same contextual knowledge that the human teacher has about the task, and thus does not know which object features are relevant to successfully repeating the task. For example, in transferring a *cup-stacking* task, the robot would need to identify which cups in the source and target environments are similar according to their size feature. In another task in which the robot learns to *sort* the same cups by their color, the robot would need to instead map objects according to their *hue* feature. While the human teacher knows which features are relevant in the context of that task, the robot does not.

In this chapter, we describe a human-guided approach to task-dependent object mapping (*"situated mapping"*) problems. In Section 4.1, we define the situated mapping problem. Since the human teacher knows the roles of objects in the task, we posit that human teachers can readily provide assistance in this regard. We describe *Mapping by Demonstration* (MbD), our interactive approach to situated mapping problems, in Section 4.2.

We then present the results of three experiments. In our first experiment, which we first described in [85], we perform an extensive evaluation of MbD in *simulated* domains,

applying this approach to variations of tasks containing 5, 6, or 7 objects. We then follow-up with a case study demonstrating the approach's applicability to real-world objects in realistic tasks. In both the simulated and real-world tasks in this experiment, mapping assistance is provided according to the ground-truth object mapping, and thus there is no opportunity for error in the mapping assistance provided to the algorithm.

In our second experiment, which we first described in [86], we evaluate the MbD algorithm's effectiveness in the *interactive* use case. We conduct a user study recording how a human teacher assists the robot in learning and transferring three ordered, pick-and-place tasks. We conclude from this experiment that structured interaction with the human teacher during task mapping is effective in both (i) reducing the number of interactions needed to repeat the task to less than 50% of those needed to re-learn the task, and (ii) increasing mapping accuracy and confidence in comparison to predicting an object mapping without human interaction.

The final experiment we describe is an offline evaluation of our method for confidence-based interaction, addressing the errors that arise from mapping assistance in the interactive use case (and that are not present in the first, simulated experiment). In this approach, we aim to minimize interaction errors by evaluating the amount of assistance needed to infer the object mapping. From this experiment, we conclude that a robot can use a confidence threshold to moderate the number of assistance requests it makes, *balancing* autonomy and interaction to infer an object mapping.

Chapter 2 summarizes prior work on this research problem. Overall, current approaches to object mapping assume (i) objects with the same classifications play the same role in any task, (ii) the robot knows a priori which object features to use for mapping, (iii) the robot may continue to explore/train in the target environment, and/or (iv) an abstraction of the object can be used to identify specific instances of that object symbol. However, these assumptions do not hold in situated mapping problems, where we would like the robot to receive very limited new data/demonstrations of a task and then identify an object mapping

without knowledge of the specific object features relevant to that task.

In this chapter, we present the Mapping by Demonstration (MbD) approach to situated mapping, in which an agent uses *mapping assistance* (in the form of a limited number of object correspondences) to infer the remainder of the mapping. We apply and evaluate this approach on a physical robot in order to evaluate how mapping assistance can be obtained by an interactive robot. We present an HRI study that uses the MbD algorithm, and in doing so, we (i) evaluate a mode of interaction enabling the robot to receive mapping assistance in the target domain, and (ii) demonstrate human-guided object mapping on a physical robot.

4.1 Problem: Task-dependent Object Mapping

In situated mapping, the robot must identify the mapping that maximizes similarity between source environment objects (S) and their equivalent objects in the target environment (T), as follows:

$$map(S,T) = \underset{m}{\operatorname{argmax}} \sum_{i=0}^{|S|} \delta(s_i, m(s_i,T)) \tag{4.1}$$

where $m(s_i,T)$ returns the object in T which corresponds to object s_i according to the mapping m. This strategy is dependent on a similarity metric $\delta(a,b)$ that returns a similarity score for objects a and b. However, defining this similarity metric (and the object features it considers relevant) is nontrivial. The similarity metric that is appropriate for one task may not represent the object features relevant to another task. Consider our previous example, with cups the robot learns to use in both a stacking task and a sort-by-color task. In the first task, the robot should identify an object mapping maximizing the similarity between mapped objects based on their *size* feature, whereas the second task requires objects to be mapped according to their *hue* feature.

We address this problem of *situated* (context-dependent) mapping, in which object mapping is dependent on the task being performed. The situated mapping problem has

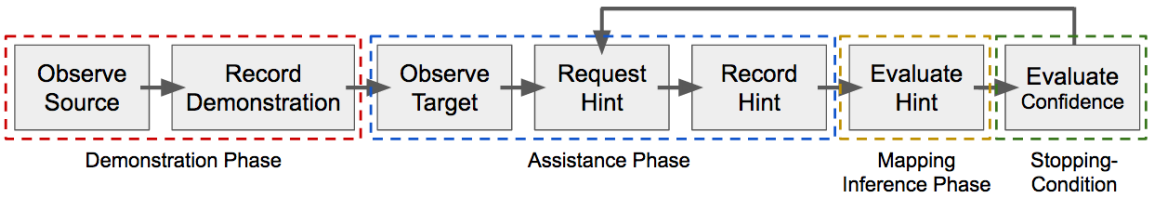


Figure 4.3: Human-guided Mapping Process

the following stages of interaction with a human teacher:

1. The human teacher demonstrates a task in the source environment.
2. The robot observes the object features and task steps involving those objects.
3. The robot observes the target environment containing new objects, and is asked by the human teacher to repeat the newly-learned task.
4. The robot infers the mapping between objects in the source and target environments.
5. The robot uses this mapping to transfer the learned task for execution in the target environment.

This chapter is concerned with how additional interaction between the robot and the human teacher can facilitate step 4 (inferring the object mapping). We currently focus on object mapping for ordered tasks where the source and target contain the same number of objects (thus requiring an *n-to-n* mapping). This lays the groundwork to address other variations of the object mapping problem in future work, including *m-to-n* mapping and partial-ordered tasks.

4.2 Approach: Mapping by Demonstration

Since the human teacher is aware of both (i) the goal of the task and (ii) the role that each object plays in achieving that goal, we propose a *human-guided object mapping* method (Fig. 4.3), consisting of two interaction phases and two mapping phases. The interaction

Table 4.1: Source of Each Object Feature's Value

Perceived Features	Derived Features	Knowledge-base Features
Location Hue Size	Spatial relations Hue-shift Size-shift	Affordances Properties

phases include a **demonstration** phase in which the robot learns the task steps from demonstration, and an **assistance** phase in which the robot records interactive assistance from the teacher. The assistance is used in the two mapping phases, first in a **mapping inference** phase, and then in the **confidence evaluation** phase which determines whether additional assistance should be requested. We now describe all four phases, later evaluating phases in isolation via simulated, interactive, and offline evaluations.

4.2.1 Demonstration Phase

At the start of the interaction, the robot observes the source environment using an RGBD sensor, and segments objects from the tabletop using the algorithm described by Trevor et al. [84]. After extracting the location, size, and hue features of each object, it derives the set of spatial relations between each object, where the spatial relations between two objects X and Y is: X above Y, X below Y, X left-of Y, and/or X right-of Y. Object size and color are used to look up *affordances* (the actions enabled by that object, e.g. *openable*, *pourable*) and *properties* (variables associated with an object's affordances, e.g. an *openable* object has the property *open* or *closed*). The robot uses a manually defined lookup table as a stand-in for a more complex process to derive this information from visual data.

In sum, these feature values are obtained from the sources listed in Table 4.1, and comprise the object representation $\langle x, y, z, c, d, s, a, p \rangle$:

- x, y, z is the centroid location
- c is the average hue of the object, ranging 0-360
- d is the bounding box volume

- $s = \{s_0, s_1, ...\}$ is the set of spatial relations between the object and all other objects, where each element $s_k \in \{$LEFT-OF, RIGHT-OF, ABOVE, BELOW$\}$
- $a = \{a_0, a_1, ...\}$ is the set of that object's affordances
- $p = \{p_0, p_1, ...\}$ is the set of property values associated with that object and its affordances

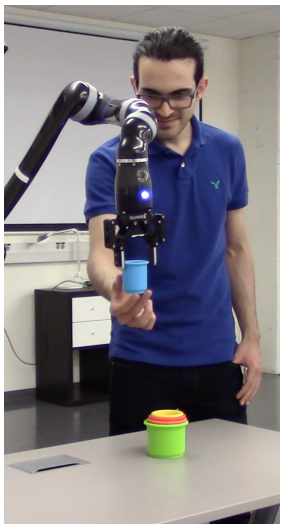
After the robot observes objects in its environment, the human teacher interacts with the robot's arm to physically guide it through executing the task (e.g. Fig. 4.4a). The robot records the trajectory of its end effector position in cartesian space, and then segments it into pick or place *task steps* according to the open/close actions of its gripper. The robot then identifies the object which was closest to the gripper at the end of each task step, recording it as the *primary object* for that task step. Following the demonstration, the task is represented as: (i) the list of object representations, and (ii) a list of task steps which indicate the primary object for each step.

In order to evaluate our algorithm in the *worst-case scenario* where the task steps contain all object features, we do not assume that the robot has learned a prior model of which object features are salient nor a model of the goal of the task, but rather include all object features in every task step. This ensures that our algorithm can complement a variety of learning algorithms that produce a list of object-directed task steps. Should the task learning algorithm also prune the list of candidate object features that are relevant for object mapping, we expect our method to demonstrate even better performance.

4.2.2 Assistance Phase

After the task demonstration, the robot may receive assistance from the human teacher to repeat the task in the target environment, later using this assistance during the mapping inference phase. Having the teacher provide assistance via natural interaction (e.g. pointing at or picking up an object) mitigates the need for human teachers to have knowledge of

(a) Demonstration Phase



(b) Mapping: Handoff

(c) Mapping: Indicate goal

Figure 4.4: Demonstration and Mapping Interactions

which features the robot has the capacity to observe (e.g. the robot can record object color, but not product brand), or how to express feature values (e.g. hue values).

Once the robot requests assistance (e.g. "Where do I go next?"), the human teacher provides a mapping assist by handing the robot the next object it should use to complete the task in the target environment (e.g. Fig. 4.4b) or, if the robot is already holding an object, by pointing to where the robot should place the object in the target environment (e.g. Fig. 4.4c). Each mapping assist indicates a correspondence between (i) the object referenced by the teacher in the target environment, and (ii) the object that *would* have been used in the *next* step in the original task demonstration.

4.2.3 Mapping Inference Phase

Two goals must be addressed simultaneously to infer the object mapping: selection of (i) an object similarity function (and associated object feature) which is representative of the mapping assistance received thus far, and (ii) an object mapping which maximizes object similarity according to the selected similarity function. As such, Algorithm 1 maintains both a (i) feature set space F containing all feature sets under consideration for the similar-

Algorithm 1 Human-guided Mapping Algorithm

1: **function** MAPPINGINTERACTION(S)
2: $T \leftarrow$ observeEnvironment()
3: $M \leftarrow$ initializeHypothesisSpace(S, T)
4: $F \leftarrow$ initializeFeatureSpace()
5: **while** target task is incomplete **do**
6: $h \leftarrow$ mapping assistance
7: $(M, F, p, c) \leftarrow$InferMapping(M, F, S, T, h)
8: **function** INFERMAPPING(M, F, S, T, h)
9: $M \leftarrow$ pruneMappingHypotheses(M, $h.src$, $h.tgt$)
10: $F \leftarrow$ pruneFeatureSpace(F, $S - \{h.src\}$)
11: $E \leftarrow$ evaluateHypotheses(M, F, S, T)
12: $p \leftarrow$ maximizeMapping(M, F, E)
13: $c \leftarrow$ evaluateConfidence(p)
14: **return** (M, F, p, c)

ity metric, and (ii) a mapping hypothesis space M containing all mapping hypotheses still under consideration. In sections 4.2.3-4.2.3, we now describe the Mapping by Demonstration (MbD) algorithm.

Initialization

The mapping hypothesis space M is initialized as the set of all possible object mappings, and thus begins as a $n!$-sized set for an n-to-n mapping problem (see Alg. 1, line 3). The feature set space is initialized as the set of object features from Sec. 4.2.1, plus two features derived from mapping assistance: size shift and hue shift (Alg. 1, line 4). dh is the average *size* shift indicated over all assistance thus far. This feature is useful for tasks in which object size is relevant to the task, but the source and target objects are at a different scale (e.g. target objects are a scaled version of source objects). ch is the average *hue* shift indicated over all assists. Similarly, this feature is useful for tasks in which object hue is relevant to the task, but differs between the two environments (e.g. blue objects in the source environment correspond to purple objects in the target). After the robot has received its first mapping assist, dh and ch are initialized to the size and hue differences, respectively, between the two objects indicated in the mapping assist, and are updated after additional

assists.

In the extensive simulated experiment (Experiment 1 described in Section 4.3), the feature set space is initialized as the power set of all object features, resulting in the initial feature set space including all 127 combinations of features as described in the original MbD implementation in [85]. We simplify this feature set space in our interactive implementation of MbD (used in Experiments 2 and 3), such that each feature set consists of a single feature. This results in an initial feature set space F consisting of only 7 feature sets, each containing one of the elements of the feature vector $\langle c, ch, d, dh, s, a, p\rangle$ consisting of hue, hue shift, size, size shift, spatial relation, affordance, and property features. A n x n x 7 evaluation matrix E is generated during initialization, containing the evaluation score for every possible object correspondence according to each of the 7 object features:

$$E_{ij} = \langle \Delta\mathbf{c}_{i,j}, \Delta\mathbf{ch}_{i,j}, \Delta\mathbf{d}_{i,j}, \Delta\mathbf{dh}_{i,j}, \Delta\mathbf{s}_{i,j}, \Delta\mathbf{a}_{i,j}, \Delta\mathbf{p}_{i,j}\rangle$$

which evaluates the correspondence between objects at indices i and j according to each evaluation metric denoted as Δx based on a single object feature x. Each evaluation metric is based on a generic distance function defined in Eq. 4.2 as the difference between two objects' value for that feature, measured along a Gaussian curve:

$$\mathrm{D}(v_1, v_2, r) = \frac{\mathcal{N}(v_2|v_1, 1\sigma r)}{\mathcal{N}(v_1|v_1, 1\sigma r)} \tag{4.2}$$

Since we cannot weight evaluation metrics based on a feature's prevalence in previous instances of that task, normalization becomes a challenge. To address this, the evaluation metric for each feature is scaled based on that feature's value range, as follows:

- *Hue similarity:* $\Delta\mathbf{c}_{i,j} = D(0, \tan^{-1}\left(\frac{\sin(c_i - c_j)}{\cos(c_i - c_j)}\right), 360)$
- *Hue-shift similarity:* $\Delta\mathbf{ch}_{i,j} = D(c_j, c_i + \mu_c, 360)$
- *Size similarity:* $\Delta\mathbf{d}_{i,j} = D\left(1, \frac{d_j}{d_i}, d_{max}\right)$

- *Size-shift similarity:* $\Delta\mathbf{dh}_{i,j} = D\left(\mu_d, \frac{d_j}{d_i}, 1\right)$
- *Spatial similarity:* $\Delta\mathbf{s}_{i,j} = D(|s_i|, |s_i \cap s_j|, |s_i|)$
- *Affordance similarity:* $\Delta\mathbf{a}_{i,j} = D(|a_i|, |a_i \cap a_j|, |a_i|)$
- *Property similarity:* $\Delta\mathbf{p}_{i,j} = D(|p_i|, |p_i \cap p_j|, |p_i|)$

where d_{max} is the ratio between the largest and smallest source objects' sizes, and μ_d and μ_c are the average differences in size ratio and hue, respectively, between object pairs in previous mapping assists. Note that this is clearly not an exhaustive list of features that may be relevant to a task; many tasks may require additional features which have not been addressed here. However, this method is intended to consider a variety of features for object mapping, and is still applicable in such tasks. Additional object features can be incorporated by defining a new evaluation metric, and expanding the evaluation matrix E to include the new metric.

After initialization, each mapping assist is used to (i) prune the mapping hypothesis space M, (ii) prune the feature set space F, and then (iii) select the highest-ranked mapping hypothesis $m \in M$ as the predicted mapping. We now describe each step in further detail.

Pruning Step

A mapping assist consists of an object in the source environment and its corresponding object in the target environment. The pruning step (Alg. 1, line 9) ensures that each remaining object mapping in the hypothesis space M contains this correspondence. A mapping hypothesis is represented as the n x n binary matrix m, where each element $m_{ij} = 1$ if $S_i \mapsto T_j$, and $m_{ij} = 0$ otherwise. The pruned mapping hypothesis space only contains mappings $m \in M$ with $m_{ij} = 1$, where i and j are the corresponding object indices in the source (S) and target (T) environments, respectively. For example, if a mapping assist corresponds source object 1 with target object 5, the mapping hypothesis space is pruned such that all remaining mappings contain $m_{1,5} = 1$. Additionally, the feature set space F is

pruned after each mapping assist (Alg. 1, line 10). A feature set is removed if it has no variance over the remaining, unmapped objects in the source environment. For example, if all unmapped objects share the same affordances, then that feature is no longer discriminating, and is irrelevant to the object mapping.

Hypothesis Evaluation Step

The evaluation matrix only needs to be generated once (during initialization). Afterward, the evaluation matrix is referenced (see Alg. 1, line 11) in order to evaluate a mapping hypothesis m according to a feature set f, as follows:

$$\mathrm{V}_{m,f} = \sum_{f_k \in f} \mathrm{sum}(m \circ E^{f_k}) \tag{4.3}$$

E^{f_k} is the n x n matrix containing the evaluation for every possible object correspondence according to feature f_k, such that every element $E^{f_k}_{ij} = E_{i,j,f_k}$. The function $m \circ E^{f_k}$ returns the entry-wise product of the object mapping m (represented as a n x n binary matrix) and the evaluation matrix based on feature f_k. This results in a matrix containing the evaluation of each object correspondence in mapping m, according to the feature f_k.

Mapping Maximization Step

Each combination of a mapping hypothesis $m \in M$ and feature set $f \in F$ is then evaluated:

$$m_0^* = \arg\max_{m \in M} \left(\max_{f \in F} \left(\frac{1}{n} \mathrm{V}_{m,f} \right) \right) \tag{4.4}$$

where n is the number of source objects, and $V_{m,f}$ is Equation 4.3. The highest-ranked mapping (m_0^*) and feature set combination is then returned as the predicted mapping (Alg. 1, line 12).

4.2.4 Confidence Evaluation Phase

A single mapping assist may not provide enough information to infer the *correct* object mapping. Thus, after receiving a mapping assist and inferring the object mapping, a decision must be made to either request additional assistance or to complete the rest of the task autonomously using the most-recently inferred object mapping. Similar to the confidence-based autonomy (CBA) approach introduced by Chernova and Veloso [69, 70], our work aims to enable the robot to rely on confidence as a means to managing its assistance from the human teacher. However, since the robot does not know which features are relevant to the task, it is unable to select features to calculate its confidence in the same manner as CBA. We propose a variation of confident execution, utilizing two sources of information available during interactive object-mapping: (1) interaction from the teacher, and (2) the resulting mapping hypothesis evaluations.

While the MbD algorithm only returns a mapping prediction after each assist, we propose that the evaluation matrix E can be used to determine the confidence of that predicted mapping (Alg. 1, line 13). We now calculate confidence based on two feature sets:

- α is the feature set leading to the highest mapping evaluation:

$$\alpha = \arg\max_{f \in F} \max_{m_0^* \in M} \mathbf{V}_{m_0^*, f} \tag{4.5}$$

  The resulting mapping is denoted as m_0^*.

- β is the feature set leading to the second-highest mapping evaluation such that $\alpha \neq \beta$ and the resulting mapping $m_1^* \neq m_0^*$:

$$\beta = \arg\max_{f \in F \setminus \{\alpha\}} \left(\mathbf{eqv}(m_1^*) \cdot max_{m_1^* \in M} \mathbf{V}_{m_1^*, f} \right) \tag{4.6}$$

$$\text{eqv}(m) = \begin{cases} 1 & \text{if } m \neq m_0^* \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

This resulting, second-highest mapping is denoted as m_1^*.

As more assistance is received, it should more clearly support one mapping hypothesis over the rest. If it does not, then the assistance provided so far supports multiple mapping hypotheses, and additional assistance is necessary to arbitrate between the top hypotheses. Thus, we use decision margin as a proxy for confidence; the more separated the top-ranked mapping is from the remaining mapping hypotheses, the more confidently it can be selected as the correct mapping. We define **confidence** as the decision margin between these two top-ranked mapping hypotheses' evaluations: $c = \text{V}_{m_0^*,\alpha} - \text{V}_{m_1^*,\beta}$.

4.3 Experiment 1: Simulated Evaluation

We first perform an extensive evaluation of the *mapping inference* phase (Section 4.2.3) in simulation. Rather than obtain mapping assistance from interaction with a teacher, the system receives assistance for each task step based on the ground-truth mapping.

We evaluated the system with three categories of simulated tasks: containing $n = 5$, $n = 6$, or $n = 7$ objects in the source and target environments. These categories represent incrementally more difficult problems; as the number of objects increases, the mapping hypothesis space from which the system must choose a single mapping increases factorially. For each category of problem, 10 mapping problems (each representing a task and consisting of a source and target environment) were generated randomly, such that each object's perceived and knowledge-base features had random value assignments, with derived features automatically generated from perceived features. To simulate how some objects may be present in both the source and target environments (as in a realistic mapping problem), each source object had a 50% likelihood of being reused in the target environment, with

the remaining objects being randomly generated. Reused objects retained intrinsic feature values (size, color, and affordances), but were given a randomly assigned location and properties, since the values of these features can be changed without replacing the object (e.g. moving a cup to a new location or emptying a cup so that its "is_filled" property changes).

Finally, ground truth mappings for each task were generated corresponding to feature sets consisting of one or two features (except for redundant feature sets {size, size-change} and {hue, hue-change}), resulting in a set of 26 possible ground truth mappings. The source and target environments had the same number of objects, so a bijective mapping was generated. This resulted in a total of 780 evaluations (3 categories x 10 tasks x 26 ground truth mappings). Note that this does not result in 780 *unique* ground truth mappings, since two feature sets may result in the same ground truth mapping.

Mapping assistance was also generated for each mapping problem instance. In a realistic mapping problem, one assist will be provided for each step of the task as described in Section 4.2.2. As a result, the assistance ordering will be dictated by the order in which objects are used in the task plan. Since the task plan is undefined in our simulated tasks, we evaluated the MbD algorithm using every possible assistance ordering to observe the impact of this ordering on mapping performance. This results in evaluating over a permutation of ${}^{n}P_{n-2}$ potential assistance orderings. The ordering of the last two assists does not matter, since assist $n-1$ leaves only one object remaining, resulting in the complete ground truth after $n-1$ assists.

We ran the simulated evaluation as follows:

1. Problem instances, ground truths, and assistance orderings are generated a priori.

2. For each problem instance, the system iteratively retrieves the next assist to be provided by the teacher to the object mapping algorithm and checks its predicted solution.

3. If the predicted mapping is correct, the system halts and records the number of assists

needed to get the correct solution using this assistance ordering. If the prediction is incorrect or if multiple predictions are returned, the system repeats the process by retrieving the next assist.

Results For each class of n-object problems, we collected data on the algorithm's performance over all possible assistance orderings, where we measure performance as the number of correct mappings after each assist is provided. For all of these problems, the full ground truth mapping is provided at $n-1$ assists, since only one source and target object remains to be mapped after assist $n-1$. Figures 4.5 - 4.7 compare the expected performance of the MbD algorithm over all assistance orderings, with error bars denoting one standard deviation, to two baselines: (i) expected performance when selecting a random mapping without utilizing mapping assistance, and (ii) expected performance when using mapping assistance to *only* prune the hypothesis space (described in the Pruning step in Section 4.2.3), and then choosing a random mapping from the remaining hypothesis space (rather than using the assistance to infer features on which mapping is based).

4.3.1 Real-World Case Studies

We have provided simulation results as a systematic analysis of the approach. The following real-world examples provide case studies demonstrating example tasks for which situated mapping problems exist and how they can be addressed using the MbD approach. As an initial evaluation of the suitability of the MbD algorithm for a robot learner, we tested it on two physical tasks: a dish sorting task (shown in Figure 4.8(a)) and a stacking assembly task (shown in Figure 4.8(a)). The robot passively observed its environment, and did not record or execute any actions. Similar to the simulated evaluation, we ran the real-world evaluation as follows:

1. The mapping ground truth(s), task plan, and object affordances/properties are defined a priori. In the sorting task, there are several correct object mappings, whereas the

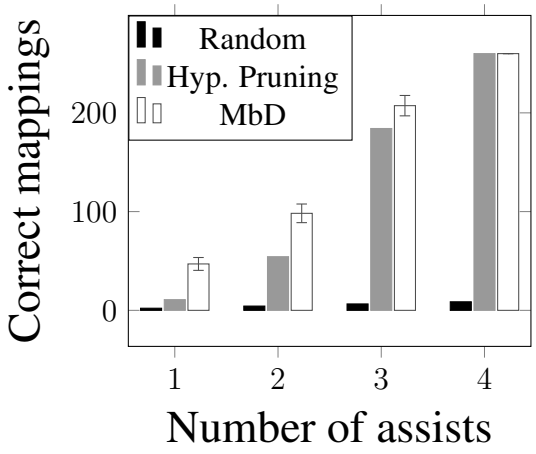


Figure 4.5: Performance in 5-Object Tasks

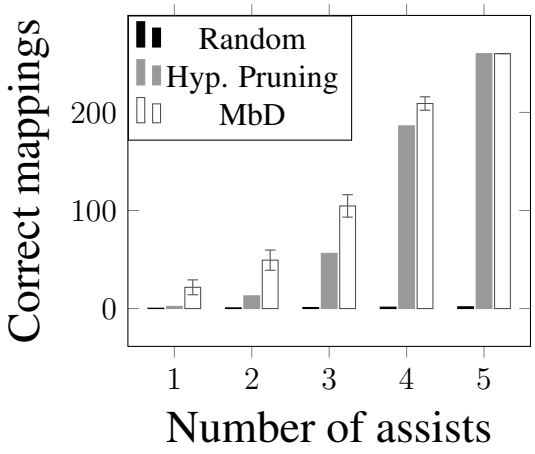


Figure 4.6: Performance in 6-Object Tasks

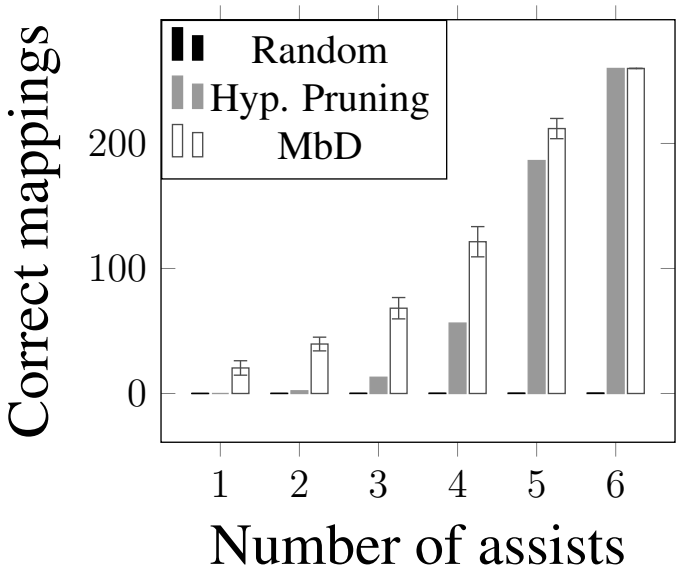
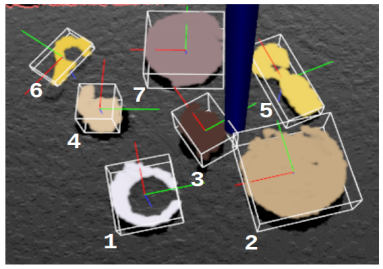
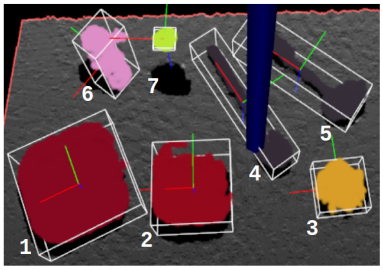


Figure 4.7: Performance in 7-Object Tasks



(a) Sorting Task Source Environment

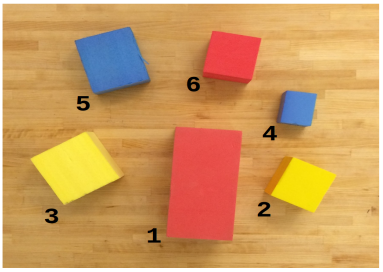


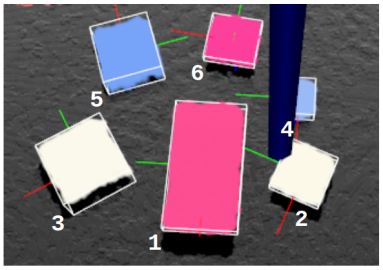
(c) Sorting Task Target Environment

```
clean dish 1
put dish 1 away
clean dish 4
put dish 4 away
clean dish 6
put dish 6 away
clean dish 3
put dish 3 away
clean dish 7
put dish 7 away
clean dish 2
put dish 2 away
clean dish 5
put dish 5 away
```

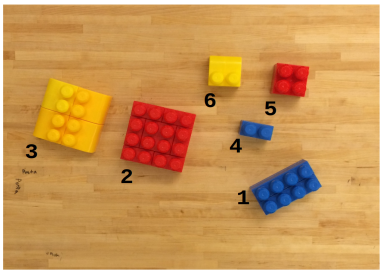
(b) Sort Plan

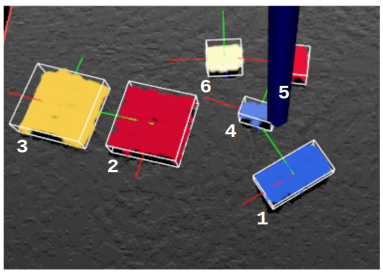
Figure 4.8: Sorting Task Environments





(a) Assembly Task Source Environment





(c) Assembly Task Target Environment

```
pick up part 1
install part 1
pick up part 3
install part 3
pick up part 5
install part 5
pick up part 6
install part 6
pick up part 2
install part 2
pick up part 4
install part 4
```

(b) Stacking Plan

Figure 4.9: Assembly Task Environments

assembly task has a single correct object mapping.

2. We set up the source environment and recorded the robot's observation of the scene, repeating this step for the target environment.

3. The algorithm accepts the assist corresponding to the next object used in the task plan and predicts a mapping solution.

4. If correct, the algorithm halts and records the number of assists needed to get a correct solution using this assistance ordering. If the prediction is incorrect, the algorithm repeats the previous step and accepts the next mapping assist.

Rather than generate feature values as in the simulated evaluation, in this evaluation all objects' perceived features listed in Table 4.1 are observed from an RGBD sensor above the environment. We incorporate perception to provide an example of how our mapping strategy may be applied to real-world problems, but do not consider the perception aspect itself to be a contribution of our work. Our main focus is on the underlying mapping strategy, and thus we indicate the perceptual features which our mapping algorithm uses and identify sources from sensors and a knowledge base.

Objects are perceived by abstracting a set of segmented objects from the point cloud using the algorithm described by Trevor et al. [84]. A bounding box is fitted to each segmented object to approximate its centroid $\langle x, y, z \rangle$ and dimensions $\langle width, depth, height \rangle$ as shown in Figure 4.8(a). The object's overall hue is derived from average hue of each pixel located at the surface of the object. Once perceived features are obtained, object locations and dimensions are used to derive a set of spatial relations between objects. Finally, size and color are used as a heuristic to assign an ID to each object, which is then used to retrieve its knowledge-base features: the affordances and properties associated with that object. Currently, we manually provide the affordances and properties associated with each object ID. In future work, we plan for the robot to autonomously retrieve this knowledge using an object classifier.

Table 4.2: Provided Object Knowledge Base

Object	Affordances	Properties
Cups	Fillable, Pourable	Empty, Full, Upright
Bowls	Fillable, Stackable	Empty, Full, Upright
Utensils	Scoopable, Pourable	Empty, Full, Upright
Blocks	Stackable	N/A

Cleaning and Sorting Task

In the first task, each environment consists of two utensils, two cups, and three bowls, which are to be cleaned and sorted separately based on their object type (indicated by their affordances). The source and target environments are shown in Figures 4.8(a) and 4.8(c), respectively. Table 4.2 lists the affordance and property values provided for these objects. The remaining features listed in Table 4.1 are derived from perception.

In contrast to simulated evaluations, there are several correct mappings for this task, since any bowl in the source environment can be mapped to any target bowl, either source utensil can be mapped to either target utensil, and either source cup can be mapped to either target cup. Whereas the task plan was undefined in the simulated evaluations, and thus the algorithm was evaluated over every assistance ordering, we use the task plan shown in Figure 4.8(b) to define the assistance ordering. Assistance was provided in the order in which objects would be used to complete the sorting task in the source environment: starting with the object closest to the robot's left hand, and continuing in order of increasing distance from the robot's hand.

The algorithm returned eight mappings (all of which were correct for the task) after the first assist: the correspondence between the small white bowl and small yellow bowl. This assist, and one of the returned mappings, is listed in Table 4.3.

Assembly Task

In the second task, each environment consists of six colored blocks of various sizes. The goal of this task is to assemble a model by stacking the blocks in a repeating color sequence

Table 4.3: Predicted Mappings After Each Assist

Task	Assist	Predicted Mapping
Sorting	$\mathbf{1 \mapsto 3}$	$1 \mapsto 3$**,** $2 \mapsto 1$**,** $3 \mapsto 6$**,** $4 \mapsto 7$**,** $5 \mapsto 5$**,** $6 \mapsto 4$**,** $7 \mapsto 2$
Assembly	$1 \mapsto 2$	$1 \mapsto 2$, $2 \mapsto 6$, $3 \mapsto 3$, $4 \mapsto 1$, $5 \mapsto 4$, $6 \mapsto 5$
	$3 \mapsto 3$	$1 \mapsto 2$, $2 \mapsto 6$, $3 \mapsto 3$, $4 \mapsto 1$, $5 \mapsto 4$, $6 \mapsto 5$
	$\mathbf{5 \mapsto 1}$	$1 \mapsto 2$, $2 \mapsto 6$, $3 \mapsto 3$, $4 \mapsto 4$, $5 \mapsto 1$, $6 \mapsto 5$

(red-yellow-blue) and in order of decreasing size (such that large blocks are placed first). Thus, objects should be mapped according to their hue and relative size. The source and target environments each contain a different kind of block, and are shown in Figures 4.9(a) and 4.9(c). Affordance and property values were provided as listed in Table 4.2, and remaining features were derived from perceptual information.

As in the cleaning and sorting task, object assistance was provided in the order in which objects would be used to complete the task in the source environment. Objects in this task are stacked in order of the required color sequence and in decreasing size as listed in Figure 4.9(b) (the task plan referencing objects in the source environment); thus, the object corresponding to the large red block was provided first, then the object corresponding to the large yellow block, and so forth. There is one correct mapping for this task, which was returned by the algorithm after three assists. Each of the provided assists and the predicted mapping after each assist is listed in Table 4.3, with the bolded assist being the final one provided before the algorithm returned the correct mapping.

4.3.2 Implications of Simulated Evaluation Results

In any mapping problem, the number of possible object mappings increases factorially with the number of objects present. Graph isomorphism is an intractable problem in general, and thus this attribute is inherent to any technique for object mapping. This motivates situating mapping in the task environment. While all mapping hypotheses are considered, leveraging mapping assistance helps (i) prune the hypothesis space after each assist, (ii) prune the feature set space, and (iii) re-evaluate the remaining mapping hypotheses to produce a

mapping prediction.

The results (Figures 4.5 - 4.7) indicate that using mapping assistance increases the robot's likelihood of predicting a correct mapping quickly. Even when mapping assistance is only used to prune the hypothesis space, as shown in the Hypothesis Pruning results in the simulated evaluation, the robot's likelihood of selecting a correct mapping is dramatically increased over that of choosing an object mapping at random. The MbD algorithm provides further benefit by inferring additional information from the assistance; rather than only use the mapping assist to prune the hypothesis space, it is also used to infer the feature(s) on which mapping may be performed. This benefit is especially evident as the hypothesis space increases. Particularly, Figure 4.7 shows that in the 7-object task (the category of mapping problems with the largest hypothesis space), the MbD algorithm correctly solves significantly more problems within the first 1-4 assists than either the Hypothesis Pruning or Random Mapping baselines.

The two perceived tasks (sorting and assembly) demonstrate the use of MbD on physical tasks such as those a robot would need to encounter. By assisting the robot with the initial steps of a task in the target environment, the robot would be able to complete the rest of the task autonomously once it has inferred a correct mapping. The cleaning and sorting task described in Section 4.3.1 would consist of at least two steps per object (clean dish x, put dish x away), resulting in a task containing a total of 14 steps. Only one mapping assist was needed for the robot to predict a correct mapping; as a result, the robot requires mapping assistance only during the first step of the cleaning and sorting task (shown in bold in Figure 4.8(b)), and would be able to execute the remaining 13 steps autonomously. Similarly, the assembly task described in Section 4.3.1 would consist of two steps per object (pick up part x, install part x), resulting in a task containing 12 steps. Three mapping assists were needed for the robot to predict the correct mapping; as such, the robot requires assistance only during the first five steps of the assembly task (shown in bold in Figure 4.9(b)), and would be able to execute the remaining seven steps autonomously.

Scaling Considerations

Two variables affect the scalability of the MbD algorithm: the number of objects that are to be mapped, and the number of features that are considered when evaluating each mapping hypothesis. As Figures 4.5 - 4.7 indicate, as the number of objects to map increases, more assists are needed in order to converge on the correct mapping hypothesis. We expect this trend to continue for problems containing $n > 7$ objects. Additionally, the initialization of the n x n x 7 evaluation matrix E (described in Sec. 4.2.3) will increase factorially with n, and linearly with the number of features (in our evaluation, 7 features were used). Following initialization, this matrix is only updated for relative features (e.g. hue-shift and size-shift similarity).

In a practical application of this algorithm, we expect its scalability to be most challenged in noisy real-world environments, when there are objects that are visible to the robot (and thus increase the complexity of the mapping problem) but irrelevant to the task. For example, a robot that is sorting dishes in a kitchen cabinet is likely to observe other kitchen objects that are unrelated to the sorting task. The MbD algorithm could still be applied to these types of scenarios by considering correspondences between irrelevant objects in its mapping hypotheses; however, this would be an inefficient application of the algorithm. A more efficient approach would be to separate the filtering problem from mapping, using a more suitable algorithm for object filtering so that mapping can be performed only on objects known to be relevant to the context of the task. In the sorting task scenario, this would result in only dish objects remaining after performing the filtering step, such that mapping can then be performed within the context of objects relevant to the dish sorting task.

While we have limited the scope of this work to n-to-n mapping, this filtering problem also poses an example of how some m-to-n mapping problems may be reduced to an n-to-n problem. In this example, the $m - n$ additional objects are distractor objects that are not relevant to the task, and thus should not be included in the object mapping. Filtering these objects prior to mapping reduces it to an n-to-n problem. Similarly, for m-to-n problems

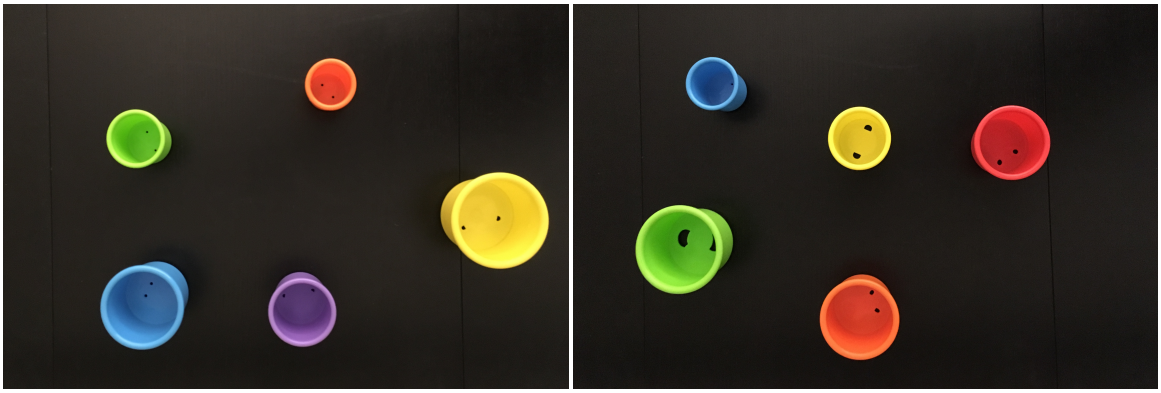
in which multiple objects in the source environment map onto a single object in the target environment (or vice-versa), objects may be clustered prior to mapping to reduce it to an n-to-n problem. An additional approach, albeit potentially computationally-expensive, would be to perform mapping for each n-to-n subset of the original m-to-n problem, selecting the mapping which results in the best evaluation score overall. Since the cause of the additional objects is contextual (such as whether it is due to the presence of distractor objects or object composition/decomposition), we leave this problem of m-to-n mapping to future work.

Finally, these results are obtained using simulated assistance, and thus relies on the assumption that mapping assistance is always correct. In a realistic interaction, however, the robot would need to obtain this assistance from the human teacher. This introduces several potential sources for error, such as mis-interpretation of the human teacher's assistance, or the teacher's mis-interpretation of the task. In the next section, we evaluate the MbD algorithm in the interactive context, and explore the effects of interaction error on the algorithm's performance.

4.4 Experiment 2: User Study Data Collection

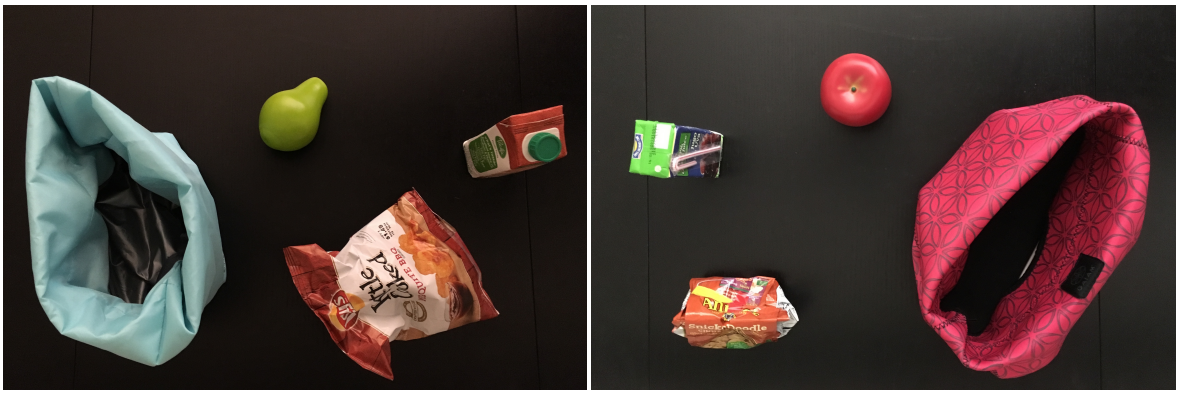
We now explore the interactive use case of the MbD algorithm. We collect interaction data from a user study consisting of the *demonstration* and *assistance* phases described in Section 4.2, collecting mapping assistance from the study participant for *every* step of the task. We later perform an offline evaluation of the remaining two phases of our approach, *mapping inference* and the *confidence-based stopping condition,* in Sections 4.4.3-4.5. Collecting the participants' assistance with every step of the task allows us to later analyze (i) whether the mapping assistance provided by participants could be used to infer the correct object mapping, (ii) the minimum number of assistance requests which were necessary to correctly infer the object mapping, and (iii) the robot's ability to evaluate the confidence of its inferred object mapping.

Eleven participants were recruited from a university campus, each teaching the robot



(a) Stacking: Source and Target Environments

(b) Sorting: Source and Target Environments



(c) Lunch: Source and Target Environments

Figure 4.10: Objects in source and target environments for each task. Figures are best viewed in color.

Table 4.4: Comparison of User Study Tasks

Task	# of Objects	# of Mapping Hypotheses	# of Correct Solutions	# of Steps	Mapping Type	Source/Target Env. Differences	Affordances	Properties
Stacking	5	120	1	8	Size	Same object set	{pourable, stackable}	{upright}
Sorting	6	720	8	6	Color	Different object set between source/target	{pourable, stackable}	{upright}
Lunch-Packing	4	24	1	6	Affordance	Different object sets within environments	{fillable, openable, edible, pourable}	{open, closed, full, empty, upright}

three pick-and-place tasks (as in Figure 4.4). A recording error occurred during one participant's interaction, and so data from 10 of the participants (8 male, 2 female) could be utilized. To begin, we guided each participant through moving the robot's arm to a series of poses to gain familiarity with the arm's joint configuration, and then through demonstrating an example pick-and-place task to gain familiarity with trajectory demonstrations. The robot perceived its workspace with a RGBD camera, and received demonstrations and manipulated objects using a 6 degree-of-freedom Kinova Jaco2 arm and a Robotiq-85 gripper. A Wizard of Oz interface was used for gesture interpretation, indicating to the robot which object the participant handed to it, and which object the participant then pointed to. A similar interface was used for speech recognition to toggle opening/closing the robot's gripper when verbally indicated by the participant.

4.4.1 Obtaining Demonstrations and Assistance from Interaction

In the first phase, the *demonstration phase*, the robot first observed its environment, recording the features of all objects present. Participants then physically guided the robot's arm to complete the task (as shown in Figure 4.4a), verbally indicating when the robot should open or close its gripper. After the demonstration, the robot's arm was placed into a tucked pose while the workspace was cleared.

In the second phase, the *assistance phase*, the tabletop contained the target environment objects to be used to repeat the task. The robot again recorded the features of all objects present. After observing the environment, the robot moved its arm to a location centered above the workspace (shown in Figure 4.4b), asking "What do I use now?". Participants were instructed to place the first object used for the task in the robot's gripper. After grasping the object, the robot asked the participant "Where do I go next?", to which participants were to respond by pointing at the location where the object was to be placed (as shown in Figure 4.4c). The robot then placed the object at the indicated location, and then returned to the central location shown in Figure 4.4b. While not all mapping assists are necessary for the robot to infer the correct object mapping, we recorded mapping assistance for all steps of the task in order to evaluate performance after *each* assist; as a result, the process of requesting an object, requesting a goal location, and placing the object was repeated until all task steps were completed successfully.

4.4.2 Evaluation Tasks

Participants performed both the demonstration and assistance phases for three fully-ordered pick-and-place tasks. These task explore several variables outlined in Table 4.4. In a **stacking task**, participants were instructed to teach the robot to stack the source and target objects (left and right images in Fig. 4.10a, respectively) one at a time. Objects in the source and target environment were obtained *from the same object set*, and thus had similar feature values (e.g. object dimensions and hue). In a **sorting task**, participants were to teach

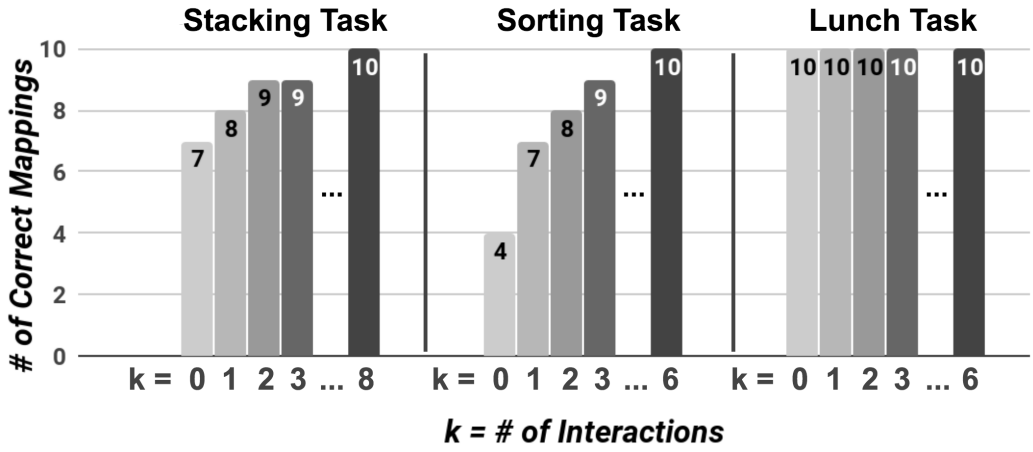


Figure 4.11: Number of correct mappings inferred within n assists for each task. Performance at 0 assists represents the unguided baseline, and maximum # of assists (k=6 or k=8) represents the task relearning baseline. In 9 of 10 stacking tasks, assistance was needed for *at most* 2 steps (1/4 of the total number of task steps) before the correct mapping was inferred. In 9 of 10 sorting tasks, assistance was needed for at most 3 steps (1/2 of the total number of task steps). In the lunch task, assistance was not necessary to correctly infer the mapping in all 10 task instances.

the robot to sort cups into a red, green, and blue stack (in that order), and then assist the robot in sorting bowls in the same order (Fig. 4.10b). The source and target objects were obtained *from different object sets* (cups and bowls). This task has eight possible correct mappings, since either blue cup may be correctly mapped to either blue bowl, either green cup may be mapped to either green bowl, and either red cup may be mapped to either red bowl. Finally, in a **lunch-packing task**, participants were to teach the robot to pack the drink, fruit, and stack items into the lunch bag (Fig. 4.10c), in that order. Later, participants were to assist the robot in packing the second set of lunch items in the same order. This task uses the most realistic objects, where objects within each environment were obtained *from different sources* (and thus the most likely to differ in their perceptual features). The initial object configurations were varied such that the initial configuration of objects in the source and target environments differed from one participant to the next.

4.4.3 Evaluating Interactive Mapping on Physical Robot

After recording all mapping assists during the assistance phase, we provided each assist to the object mapping algorithm incrementally in an offline evaluation. We evaluated performance according to **two criteria**: (1) *correctness* at assist k was measured based on the number of mappings correctly predicted with $\leq k$ assists, and (2) *interaction efficiency* was measured as the number of interactions needed to infer the correct mapping. We compared the results of the human-guided mapping approach to **two baseline methods**: (1) *"unguided mapping"* (Algorithm 2) where object mapping is performed without mapping assistance, and (2) *"task relearning"* where we record the theoretical performance if the teacher were to repeat the task demonstration in the target environment. We use this second baseline as a performance upper bound, assuming that relearning the task would require the same number of steps as was recorded in each participant's original task demonstration, and that it would result in perfect execution of the task in the target environment.

Algorithm 2 Unguided Mapping Algorithm

```
1: function MAPPINGINTERACTION(S)
2:     T ← observeEnvironment()
3:     M ← initializeHypothesisSpace(S, T)
4:     F ← initializeFeatureSpace()
5:     F ← pruneFeatureSpace(F, S)
6:     E ← evaluateHypotheses(M, F, S, T)
7:     p ← predictMapping(M, F, E)
8:     return p
```

Object Mapping Results

We first identify the worst-case performance of human-guided object mapping; that is, for each task, we find the upper-bound k number of assists such that the algorithm's best performance is reached with $0 \leq n \leq k$ assists on all instances of that task. These results are recorded in Figure 4.11. The unguided mapping baseline is represented as the results at 0 assists. The results of human-guided mapping is represented as its performance within

Table 4.5: Interaction Results

Task	# of demos in expected order	# of assists in expected order	# of consistent demos & assists
Stacking	6/10	7/10	**4/10**
Sorting	6/10	6/10	**5/10**
Lunch	8/10	10/10	**8/10**

assists 1-3; results after assists 3 are not shown because performance did not improve after the third assist. Task relearning is represented as expected performance after the number of steps that would need to be demonstrated for the task (6 or 8 steps, depending on the task).

Efficiency Implications Figure 4.11 indicates that the human-guided method achieved its highest performance for the lunch-packing task without any additional assistance, highest performance for the stacking task with **at most two** assists, and highest performance for the sorting task at most **three** assists. Its performance for the stacking and sorting tasks resulted in approximately 1.3x and 2.25x as many correct mappings, respectively, as the unguided method. For the lunch-packing task, the unguided mapping algorithm was able to perform perfectly, since there was a clear similarity between the objects' affordance and property features in the two environments, and little similarity between other features.

Overall, these results indicate that a robot using the human-guided mapping algorithm in real-time would require only modest assistance to infer the correct mapping and repeat the rest of the task autonomously. For 9/10 instances of the stacking task (consisting of 8 steps) the robot would only need assistance with *at most* 25% of the task (2 steps) to target the correct objects *autonomously* in the remaining 75% of the task. Similarly for 9/10 instances of the sorting task (consisting of 6 steps) the robot would need assistance with the first 50% of the task (3 steps) *at most* to target the remaining objects autonomously. Note that in a typical case, the robot would need even fewer assists (discussed in Sec. 4.5).

Interaction Results

Next, we specifically analyze the human teacher's *interactions* with the robot during the demonstration and assistance phases. While the results in Sec. 4.4.3 indicate that the mapping algorithm is able to efficiently infer the correct object mapping from assistance, there were several instances in which the teacher did not provide the expected assistance. Table 4.5 lists the number of demonstrations in which the task was recorded in the expected order in its entirety. In the remaining demonstrations, the objects used in each task step was either (i) unclear to the robot for a particular step (and thus the wrong object was recorded for that step), or (ii) provided to the robot in the wrong order by the human teacher's demonstration. Similarly, the second column lists the number of assists that were provided in the expected order during the assistance phase. The last column indicates the number of task instances in which the demonstration and assistance were provided and recorded in a consistent order. Overall, 43.3% of task instances had at least one inconsistency between the demonstration and assistance orderings, despite the task order being specified prior to the demonstration and assistance phases.

4.4.4 Effects of User Study Data on Algorithm Performance

We now discuss three implications of the inconsistencies between demonstration and assistance orderings: (1) their effect on mapping results, (2) the increased error risk incurred by interactive assistance, and (3) proposed strategies for reducing the number of inconsistent assists.

Effect on Mapping Results

There were two sources of error which led to inconsistencies between the demonstration and assistance: recording errors and interaction errors. As an example of an interaction error, the very first mapping assist for a sorting task was provided in the wrong order, after which the mapping algorithm was unable to recover despite receiving additional assistance

(still resulting in 9/10 performance at k=3 assists, compared to the hypothetical 10/10 performance expected with the relearning upper-bound baseline shown at k=6).

As an example of a recording error, one participant demonstrated the stacking task by having the robot first move the largest, yellow cup to a central location, and then move all other cups to the yellow cup in its new location. Since the objects were moved to a location which the robot did not record during the initial observation, it recorded the task steps as stacking the cups into the wrong object, leading to incorrect mapping assistance being recorded.

Additional recording errors may occur as a result of mis-classifying semantic features (e.g. object affordances and properties). While these semantic features were manually defined in our evaluation, we expect that these features could be derived from a knowledge base or visual classifier, which may introduce additional error depending on how well they are suited/trained for the target domain. In tasks where objects should be mapped based on a particular semantic feature, this error could result in an incorrect mapping being selected if that feature is mis-classified or missing from the knowledge base.

Increased Error Risk

The effects of inconsistent assistance presents a downside to relying on assistance from the human teacher for object mapping: as the robot requests more assistance, the potential for incorrect assistance increases. Figure 4.13 illustrates this tradeoff; the number of correct mappings increases (or remains stable) with the increase in the number of assists, until a point at which the additional (incorrect) assistance actually causes the number of correct mappings to *decrease*.

We observe that this tradeoff does not occur when the algorithm is performed over a "consistent dataset": the subset of data in which the demonstration and all assistance were provided and recorded in a fully-consistent order (4 instances of stacking, 5 instances of sorting, and 8 instances of lunch-packing). Rather, we observe that it correctly infers the

object mapping in all 17 of these instances, and that the algorithm performance does *not* decrease with additional assistance (in contrast to Fig. 4.13).

Thus, one method of addressing this tradeoff is to request just enough assistance for the robot to maximize mapping performance, while also minimizing the risk of receiving incorrect assistance. Our third experiment demonstrates a confidence-based threshold that aims to achieve this balance. While limiting the amount of assistance reduces the opportunity for error caused by inconsistent assistance, it does not eliminate it; interaction errors that occur early in the task would still cause the algorithm to converge quickly on the wrong mapping hypothesis.

Guidelines for Interactive Assistance

The consistent-dataset results suggest that performance could be further improved by refining the interaction to reduce the teacher's likelihood of providing inconsistent assistance. We note three factors for future work: (i) ensuring that the task constraints (including the task ordering) are clear to the teacher, (ii) maintaining consistency between the source and target task orderings, and (iii) ignoring mistaken or redundant mapping assists.

Clarifying task constraints Regarding the first point, the teacher's understanding of the robot's limitations will affect the order in which they demonstrate the task, such as the robot's inability to observe the cup being moved to another location, or attempting to pick up multiple objects while the robot's gripper can only accommodate one. One solution to this challenge is to provide a visual list specifying the order in which object should be used in order to successfully complete the task. Another solution is for the teacher to be provided with practice trials of the task in order to become more familiar with it and better understand the task ordering constraints. This would also serve to address errors caused by the teacher mistakenly repeating the task in a different order than originally demonstrated.

Reducing inconsistent assistance Providing additional transparency into the robot's reasoning process may also reduce these errors. While assistance is currently provided by the teacher via gestures, it may be beneficial for the teacher to be able to correct perceptual and/or interaction errors via another interaction method, such as speech (e.g. verbally indicating when to discard the previous assist) or a graphical interface (e.g. a visual representation of the assistance provided so far). One may also consider an alternate framework for obtaining assistance, in which the robot attempts to complete the task using its currently highest-ranked mapping hypothesis; the teacher would then provide assistance by interrupting and correcting the robot's motion, after which the robot would record the correction as an assist and re-rank its mapping hypotheses accordingly.

Accounting for noisy assistance Finally, the robot may need to account for "noise" in the teacher's assistance. When providing the robot with mapping assistance, the robot may fail to grasp the intended object, or the teacher may change his or her mind about which object should be used next in the task. In either case, the robot should record the teacher's corrective actions, rather than record the mistakes as assistance. This could be enabled by providing an interaction method for overwriting past actions or assistance (such as a speech command the teacher can use to indicate that they want to repeat a particular step of the task).

We also note that the use of assistance is intended to lower the teacher's effort in comparison to giving a full demonstration, by (i) using a simpler form of interaction (indicating an object rather than re-demonstrating a full task motion) and (ii) requiring interaction during only a subset of the task. An additional direction for future work is to explicitly measure the difference in perceived cognitive effort when providing assistance versus a full re-demonstration.

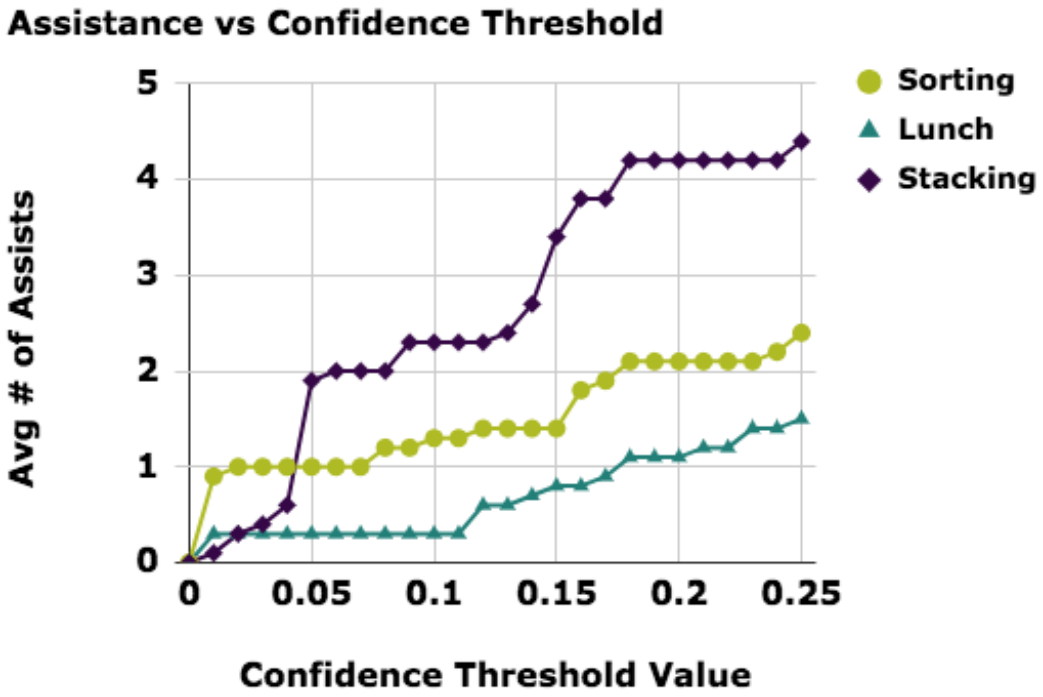


Figure 4.12: As the confidence threshold value increases, so does the number of assists needed to attain that threshold

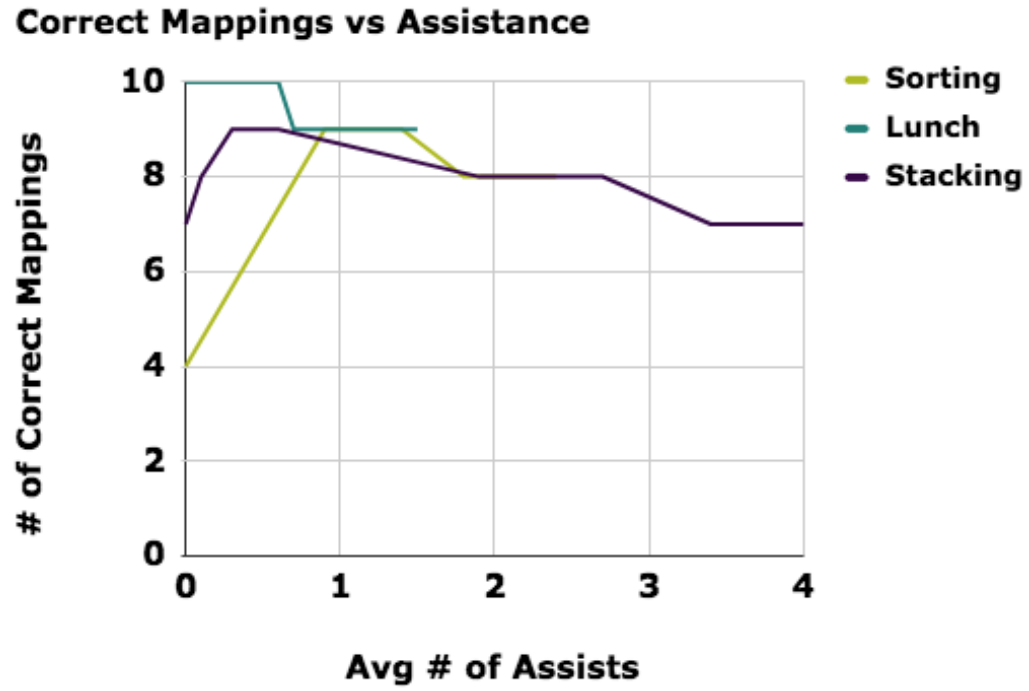


Figure 4.13: Performance initially increases after assistance, but later decreases with further assistance

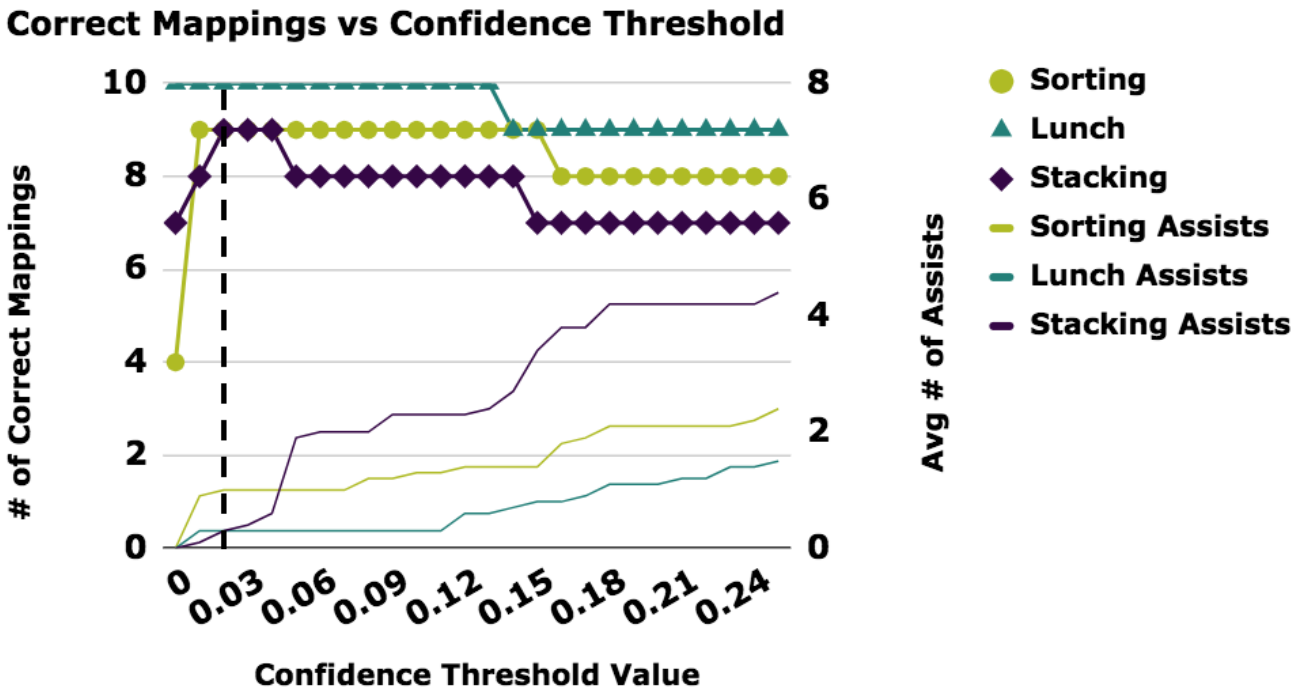


Figure 4.14: Relationship between confidence threshold (x-axis), average # of assists (bottom lines, sourced from Fig. 4.12), and performance (top lines). Threshold of 0.02 (dashed line) maximizes performance and minimizes number of assists.

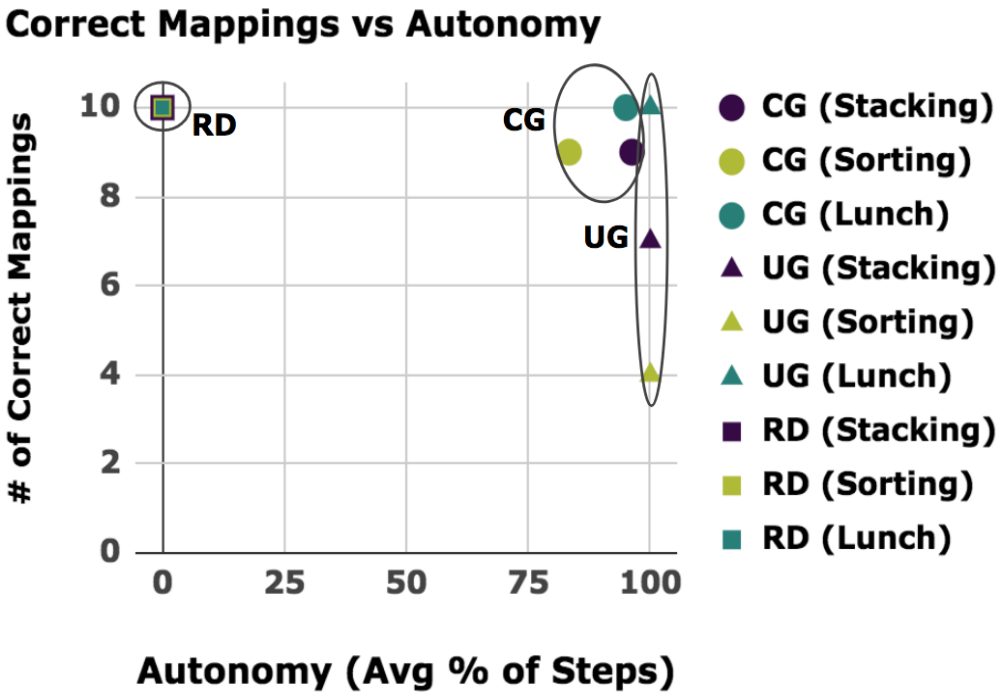


Figure 4.15: With optimal threshold value (0.02), confident guided (CG) method results in high autonomy *and* mapping correctness, compared with unguided (UG) and relearning (RD) baselines.

4.5 Experiment 3: Confidence-based Stopping Condition

The previous experiment presented a tradeoff between mapping performance and increased assistance. One solution to this tradeoff is for the robot to request just enough assistance to maximize mapping performance, while also minimizing the risk of receiving incorrect assistance. Minimizing the number of assists requested by the robot thus serves two purposes: (i) increasing the robot's autonomy, and (ii) reducing the possibility of error introduced via incorrect mapping assistance. We now evaluate the confidence-based threshold described in Section 4.2.4, considering the number of assists which are needed to maximize mapping performance when the number of assists is *variable* and based on mapping confidence (rather than identify an upper bound as in Sec. 4.4.3). Confidence values (and thus the decision margin between them) are within the range [0.0, 1.0]. We analyze performance at a range of confidence thresholds between [0.0, 0.25].

We observe the effect of confidence thresholding on mapping performance in two steps. First, changing the confidence threshold affects the number of mapping assists which are needed for the robot to reach that confidence threshold. Thus, as the confidence threshold increases, the number of requested mapping assists also increases, as seen in Fig. 4.12. The confidence threshold has a second effect: as more mapping assists are needed in order to attain the confidence threshold, the performance of the human-guided mapping algorithm also changes in response to the additional assists as shown in Fig. 4.13. Fig. 4.14 illustrates the resulting relation between the confidence threshold and the mapping algorithm's performance. The optimal confidence threshold (c=0.02) is indicated by the dashed line, and maximizes the algorithm's performance (the line charts at the top of Fig. 4.14) while minimizing the number of assists needed to attain that performance (the line charts at the bottom of Fig. 4.14). Finally, we compare the performance of (i) human-guided mapping using this optimal confidence threshold and (ii) the unguided and relearning baselines. Fig. 4.15 demonstrates how confident human-guided object mapping maximizes

average autonomy and correctness across the three tasks, whereas the relearning baseline minimizes the robot's autonomy and the unguided baseline provides fewer correct mapping results on average.

4.5.1 Autonomy and Performance Implications of Confidence-Guided Mapping

Applying this threshold to real-time object mapping would enable the robot to repeat the rest of the task autonomously. In Sec. 4.4.3, we discussed the *maximum* number of assists needed to maximize the human-guided mapping algorithm's results. However, the results from implementing a confidence-based stopping condition on our evaluation data indicate that even fewer assists are necessary in a typical mapping problem. Our results indicate that with a properly selected confidence threshold, the robot could infer the correct object mapping in 93.3% of problems and requested 0 or 1 assists in the average case, thus maximizing both *autonomy* and *correctness*. Further evaluation should test whether this confidence threshold is generalizable across a wider variety of tasks, or is dependent on a particular feature of the task (e.g. overall object similarity, or number of object features under consideration).

4.6 Summary

Without contextual knowledge about the task, such as a task-specific similarity metric for object correspondences, the robot cannot reliably identify an object mapping for task transfer. Prior work assumes that (i) the robot has access to multiple new demonstrations of the task or that (ii) the primary features for object mapping have been specified. Our method does not make either assumption; rather than requiring additional demonstrations of the task, it uses limited, structured interaction with a human teacher. Additionally, by having human teachers provide mapping assistance by indicating objects (rather than describing features), we mitigate the need for teachers to have knowledge of which features the robot can observe, or how to express feature values.

Our simulated evaluation (published in [85]) demonstrates how the use of mapping assistance quickly reduces the mapping problem complexity, enabling the correct mapping to be identified within a small number of assists. The aim of our interactive evaluation (published in [86]) was to test whether (i) participants could provide mapping assistance through natural interaction with the robot and its environment, (ii) the robot could record the objects involved in each step of the task, and (iii) the mapping assistance provided by participants could be used to efficiently infer the correct object mapping. The evaluation results demonstrate that by incorporating human interaction, our human-guided mapping approach meets these criteria, while providing mapping predictions that are accurate, confident, and obtained through a limited number of additional interactions with the human teacher. Finally, we have demonstrated how a confidence-based stopping condition can be used to moderate the robot's interaction with the human teacher, and thus find a balance between **autonomy** and **interaction**.

4.6.1 Key Contributions and Insights

This chapter presents the first algorithm that addresses task-dependent object mapping. We have introduced the Mapping by Demonstration algorithm: an interactive approach to solving situated mapping problems on a physical robot. In evaluating this approach, we have analyzed (i) how well it performs using assistance from an oracle, (ii) how it is affected by assistance obtained from interaction with human teachers, and (iii) how the robot should moderate its interaction with the teacher in order to maximize its mapping performance and autonomy. This analysis resulted in the following key findings:

Insight #1: The number of possible object mappings increases quickly with the problem size (e.g. number of objects). **As the problem space grows, it becomes increasingly important to use interaction data in multiple ways**: first by pruning the set of mapping hypotheses, and second, by inferring the task-specific similarity function that may be used to rank the remaining mapping hypotheses.

Insight #2: While **interactive assistance is effective in providing grounded mapping data, it also introduces opportunity for errors** to be introduced at several stages of the interaction (e.g. perceptual errors in recording the assistance, or mismatches in the robot's and human's task models).

Insight #3: The robot's **mapping performance is optimized when it uses a confidence-guided metric to moderate the number of assists it requests from the teacher**. This provides two benefits. First, it minimizes the number of interactions needed to confidently predict the remainder of the object mapping, and thus reduces the likelihood that one of those interactions results in erroneous data. Second, by limiting the number of interactions to those *necessary* to infer the remainder of the object mapping, the robot's own autonomy is maximized.

4.6.2 Open Questions

This chapter addresses transfer problems in which a mapping between source and target objects is sufficient to ground the task in a target environment. An underlying assumption of this work is that once the appropriate object mapping is identified, the robot can manipulate the target objects in the same way as it originally learned for the source objects. For the tasks we evaluated in this chapter (e.g. pick-and-place tasks), this assumption holds true.

However, it is easy to imagine transfer problems in which this is not the case. A new object may impose constraints that were not present with the source object, such as replacing a closed soda can with an open glass; this object replacement also introduces the constraint of keeping the glass upright, where this constraint did not exist for the original can. Additionally, an object replacement may affect the relationship between the robot's end-effector and other objects in its environment; for example, if the robot uses one object to manipulate another, any change in the first object may affect the robot's ability to manipulate the second object.

These transfer problems require not just a mapping between object labels, but also an

action mapping that dictates how the robot's trajectory execution should be adjusted to account for the new objects. In the next chapter, we consider this category of transfer problems.

CHAPTER 5

HUMAN-GUIDED TRAJECTORY ADAPTATION VIA CORRECTIONS

When transferring a task to an environment containing new objects, it is essential to consider the relationship between the new objects and the task goals. For some object replacements, such as those discussed in the previous chapter, it is sufficient for the robot to know *which* objects to manipulate throughout the task and *when* to use them. We now discuss transfer problems in which replacing one or more objects also affects *how* the robot should manipulate them. Whereas Chapter 4 introduced a method of obtaining an *object mapping* to enable transfer, we now discuss the need for an *action mapping* between the robot's trajectory when manipulating a source object and when manipulating the corresponding target object. Figure 5.1 illustrates the role of this chapter within the thesis as a whole.

A prime example of transfer occurs when replacing a tool that the robot uses to complete a task. A robot situated in human environments will encounter environments and tasks suited for human capabilities, and thus it is important for a robot to be able to use human tools [87]. While a robot can easily learn to complete a new task with a new tool via demonstrations by a human teacher, the demonstration(s) provided for that tool cannot prepare the robot for all variations of that tool it is likely to encounter. These variations can range from different tool dimensions (e.g. different sized spoons, hammers, and screwdrivers) to tool replacements when a typical tool is not available (e.g. using a measuring cup instead of a ladle, or a rock instead of a hammer). An additional challenge is that tools are often used to manipulate other objects in the robot's environment; the shape of a tool alters its effect on its environment [41], and thus a tool replacement may necessitate a change in the manipulation of that tool in order to achieve the same task goal [73].

Tools are common in human life, and thus, a robot that operates in human environments is likely to encounter situations in which it needs to use tools as well. As a result, the re-

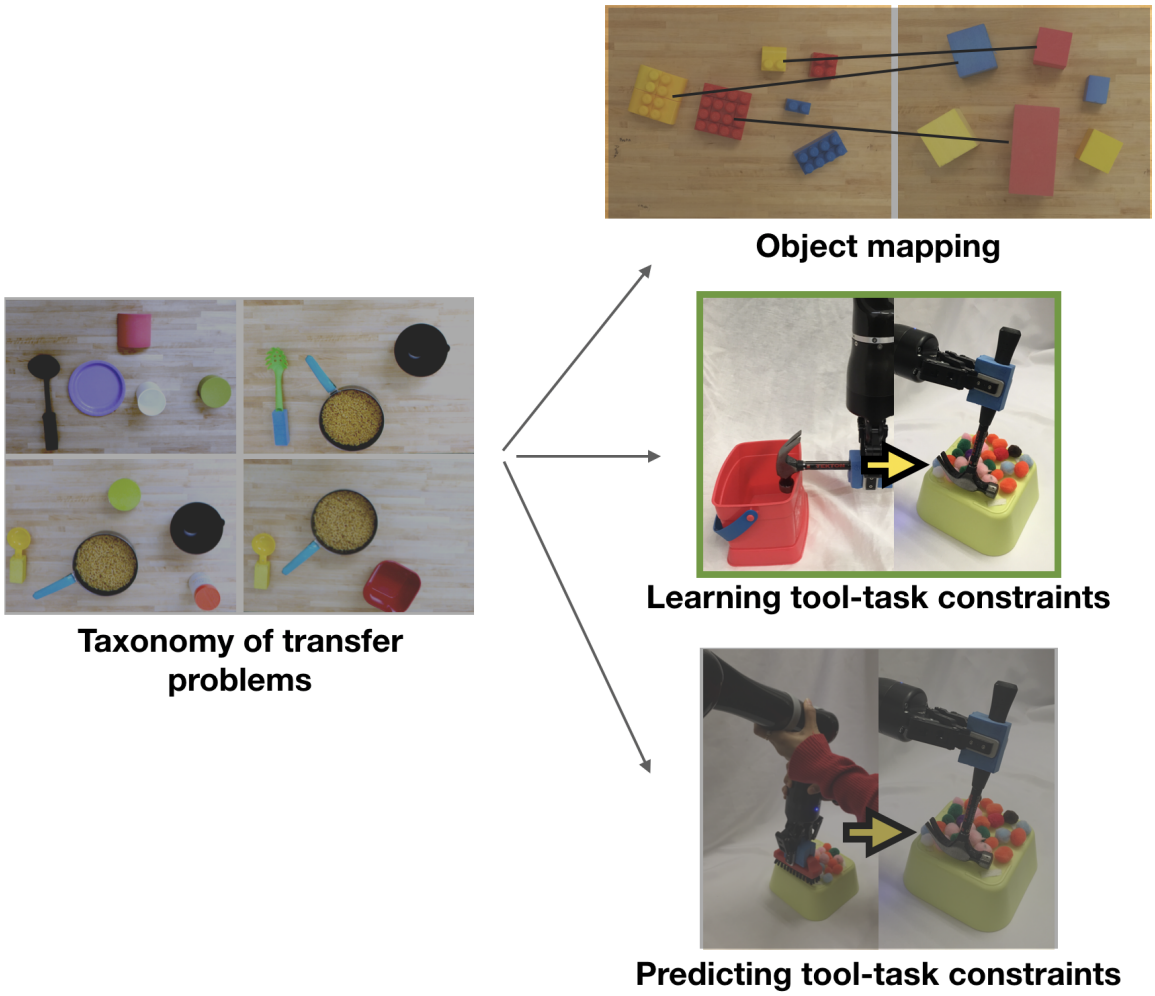


Figure 5.1: We now focus on a second category of transfer problems, highlighted in green: those in which an object replacement requires a change in the robot's trajectory in order to manipulate the new tool.

placement tool practically alters the robot's end-effector, with the exact alteration unknown to the robot.

Effects of Tool Replacement on Task Transfer Although some tools are flexible, many have a rigid structure that results in the tool affecting task execution in two ways: (1) a task is performed with respect to a particular acting surface of that tool (which we refer to as the tooltip), and (2) there is a fixed relationship between the robot's gripper and the tooltip. By learning the relationship between the robot's gripper trajectories when performing the same task with two different tools, our approach aims to learn the relationship between the tools themselves and how they are used in the context of that task. Once this relationship is learned, the robot can apply it to reuse previously-learned task models with the new tool.

While the relationship between the robot's gripper and the tooltip is static, modeling the relationship between the tooltip and the robot's motion is challenging. In general, one would expect to be able to solve this problem by modifying the trajectory so that the tooltip of the new tool followed the same path as the tooltip of the original tool. However, there are several key challenges to this approach. First, the problem of identifying the tooltip is non-trivial due to ambiguities, such as how any point of a cup's rim may be the tooltip for a pouring action. Second, a different tooltip may be used depending on the task, such as how the rim of a ladle is used for scooping, while the back of the scoop can be used to push or pull another object away. As a result, the action mapping learned for one class of tasks (e.g. pushing or scooping) may not be applicable to other tasks completed with the same tool. Additionally, some parts of a task are under-constrained (such as the robot moving a measuring cup toward a bowl), whereas other parts of the same task are constrained with respect to the tooltip (such as the edge of the measuring cup when pouring it).

Modeling Interactive Corrections An advantage of Learning from Demonstration is that the robot can quickly receive new demonstrations [10, 11] for a new tool variation by having the teacher physically guide it to repeat the task with the new tool. A more efficient

approach, however, would be for the robot to learn about the relationships between tools, such that this relationship can be extended to transfer multiple task models to be reused with the new tool.

Due to the unstructured nature of task demonstrations, the original and new demonstrations may vary in ways that do not reflect accommodations necessary to repeat the task using the new tool. Interactive *corrections* have been shown to be effective interface for adapting a previously-learned task model [22, 23, 24]. We leverage this form of interaction for tool transfer. In doing so, we minimize the distance between the original and corrected goal poses throughout the task, thus increasing the likelihood that these corrections reflect *only* the trajectory changes *necessary* for the new tool.

In this chapter, we introduce an algorithm for *transfer by correction*: an interactive approach to learning the relationship between tools such that tasks learned using one tool (such as in Figure 5.2a) can quickly be transferred to utilize a new tool (Figure 5.2c). The robot interacts with a human teacher to receive corrections when repeating a known task with a new tool, pausing to enable the teacher to correct the position and orientation of its gripper throughout the task (e.g. to correct a collision, or correct its location with respect to a target object). The algorithm then represents these corrections according to two *tool transform models* in order to identify the pose transformation that is most consistent across corrections.

Our results, first presented in [88], indicate that the tool transform models learned from one episode of task corrections can be used effectively to model the relationship between the source and replacement tool, enabling it to **achieve high performance in 83% of tool/task combinations**. Furthermore, we test the generalizability of the learned transformation to additional tasks (such as in Figure 5.2d), and find that the tool transform model **improves transfer performance in 27.8% of across-task evaluations**, and 41% of across-task evaluations in which the source and replacement tool share similarities in their tooltips. Overall, our work in this chapter demonstrates that (i) we can effectively model

the transforms between tools using interactive corrections, and (ii) the transform can be generalized to other tasks providing a similar context for the new tool, **without additional corrections, nor any training on those tool-task combinations**.

5.1 Problem Definition

Suppose that for a particular task, there exists a transfer function ϕ_a^b that transforms pose $\mathbf{P_a}$ using tool a into a pose $\mathbf{P_b}$ for tool b:

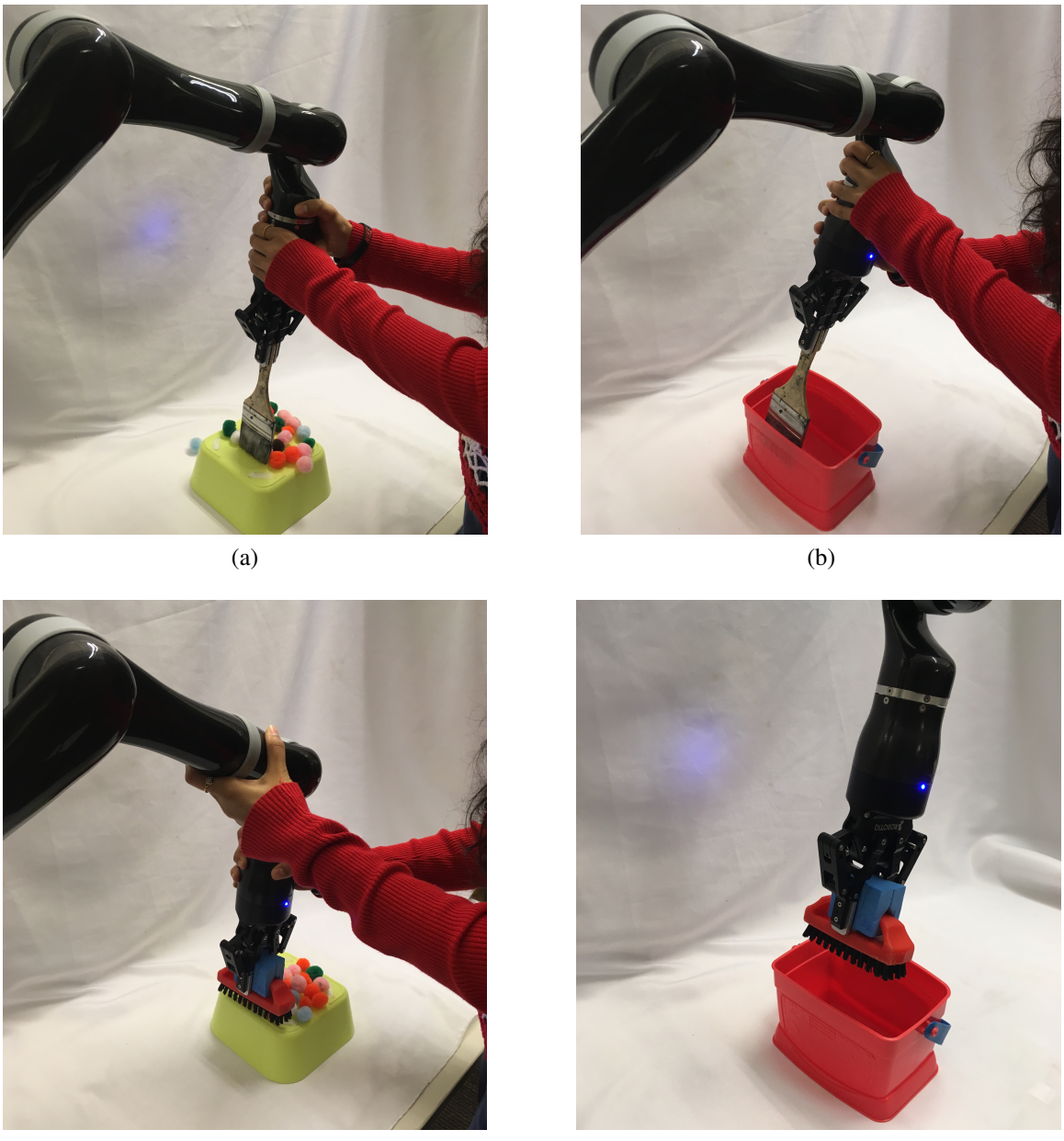
$$\mathbf{P_b} = \phi_a^b(\mathbf{P_a}) \tag{5.1}$$

We assume that each demonstration consists of a series of keyframes [13]. The robot receives corrections by executing a trajectory planned using the original task model, pausing after a time interval defined by the keyframe timings set during the original demonstration. The teacher then moves the robot's gripper to the correct position, after which the robot resumes task execution for the next time interval, repeating the correction process until the entire task is complete. Each resulting correction at interval i consists of the original pose $\mathbf{C_a^i}$ (using tool a) and the corrected pose $\mathbf{C_b^i}$ (using new tool b) at keyframe i. A collection of K corrections (one for each of K keyframes) results in a K x 2 correction matrix:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C_a^0} & \mathbf{C_b^0} \\ \mathbf{C_a^1} & \mathbf{C_b^1} \\ \cdots & \\ \mathbf{C_a^K} & \mathbf{C_b^K} \end{bmatrix} \tag{5.2}$$

Each corrected pose $\mathbf{C_b^i}$ provides a sample of the transfer function value with the original pose $\mathbf{C_a^i}$ at keyframe i as input, plus some amount of error from the optimal correction pose:

$$\mathbf{C_b^i} = \phi_a^b(\mathbf{C_a^i}) + \epsilon \qquad \epsilon_n \sim \mathcal{N}(0,\, \sigma_n^2) \tag{5.3}$$



(a) (b) (c) (d)

Figure 5.2: The robot receives demonstrations of sweeping (a) and hooking (b) tasks using the first tool (a paintbrush). After receiving corrections of the sweeping task (c) using a new tool (a short scrub-brush), the robot uses these corrections to complete an undemonstrated tool-task combination: hooking the box with the scrub-brush (d).

We assume ϵ is sampled from a Gaussian noise model for each axis $n \in [1 \dots 6]$ of the 6D end-effector pose. Our aim is to learn a transfer function ϕ that optimally reflects the task constraints, using a correction matrix $\mathbf{C}$. In this chapter, our research questions are as follows:

1. How can we learn ϕ for a particular task from $\mathbf{C}$ containing sparse, noisy corrections?

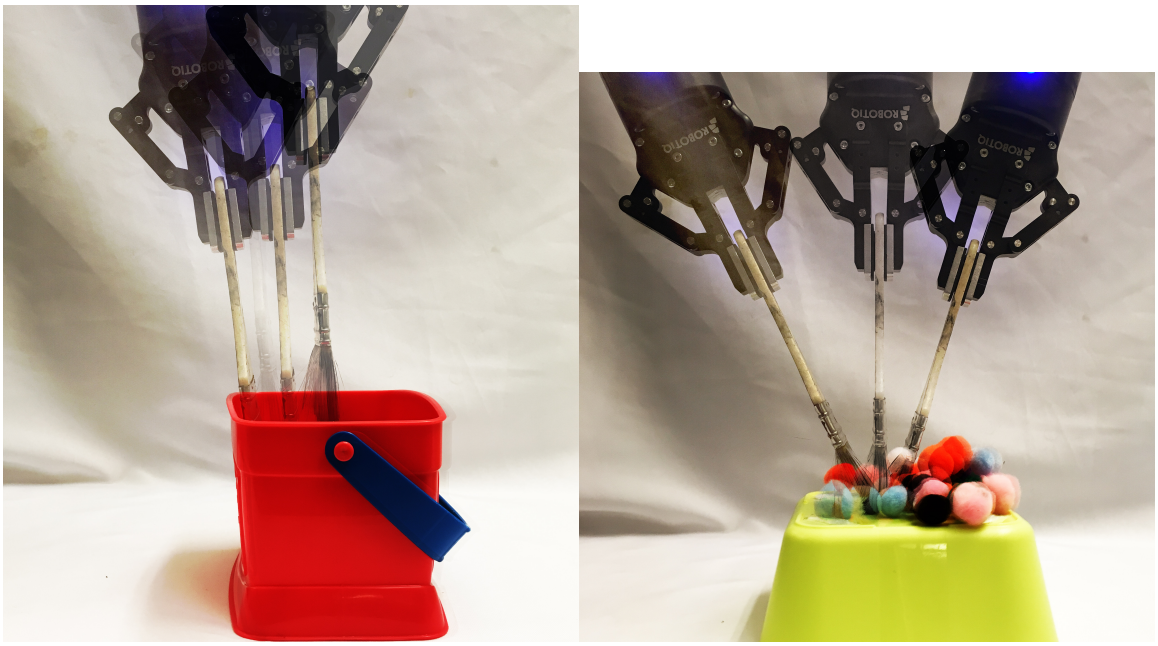
2. Under what conditions can the ϕ learned from corrections on one task be used to transfer other known tasks to the same replacement tool? What characteristics of the tool and task predict whether a previously-learned ϕ can be applied?

5.2 Approach: Transfer by Correction

Given a task trajectory $\mathbf{T}$ for tool a consisting of a series of t poses in task space such that $\mathbf{T} = [\mathbf{p_0}, \mathbf{p_1}, \dots, \mathbf{p_t}]$, we transform each pose individually for tool b. Representing an original pose for tool a in terms of its 3 x 1 translational vector $\mathbf{t_a}$ and 4 x 1 rotational vector $\mathbf{r_a}$, we transform it into a pose $\mathbf{p_b}$ for tool b as follows:

$$\mathbf{p_b} = \phi_a^b(\mathbf{p_a}) = \left\langle \mathbf{t_a} + \hat{\mathbf{t}}, \mathbf{r_a} \cdot \hat{\mathbf{r}} \right\rangle \tag{5.4}$$

Here, $\mathbf{r_a} \cdot \hat{\mathbf{r}}$ refers to the Hamilton product between the two quaternions. The goal is now to estimate the optimal rotational $\hat{\mathbf{r}}$ and translational $\hat{\mathbf{t}}$ transformation components from the corrections matrix $\mathbf{C}$, and then apply these transformations to the trajectory $\mathbf{T}$. Our approach addresses this goal by (1) modeling $\mathbf{C}$, particularly the relationship between each correction's translational and rotational components, (2) sampling a typical translational transformation $\hat{\mathbf{t}}$ and rotational transformation $\hat{\mathbf{r}}$ from this transform model, and (3) applying $\hat{\mathbf{t}}$ and $\hat{\mathbf{r}}$ to transform each pose in the task trajectory according to Equation 5.4.



(a) Orientation constraint (b) Tooltip constraint

Figure 5.3: Poses meeting the same orientation constraint share similar orientations but vary more in their position, whereas poses meeting the same tooltip constraint rotate around the tooltip.

5.2.1 Task Constraints

We observe that corrections indicate constraints of the tooltip's position and/or orientation, and that these constraints are reflected in the relationship between the translation and rotation components of each correction. Broadly, each correction may primarily indicate:

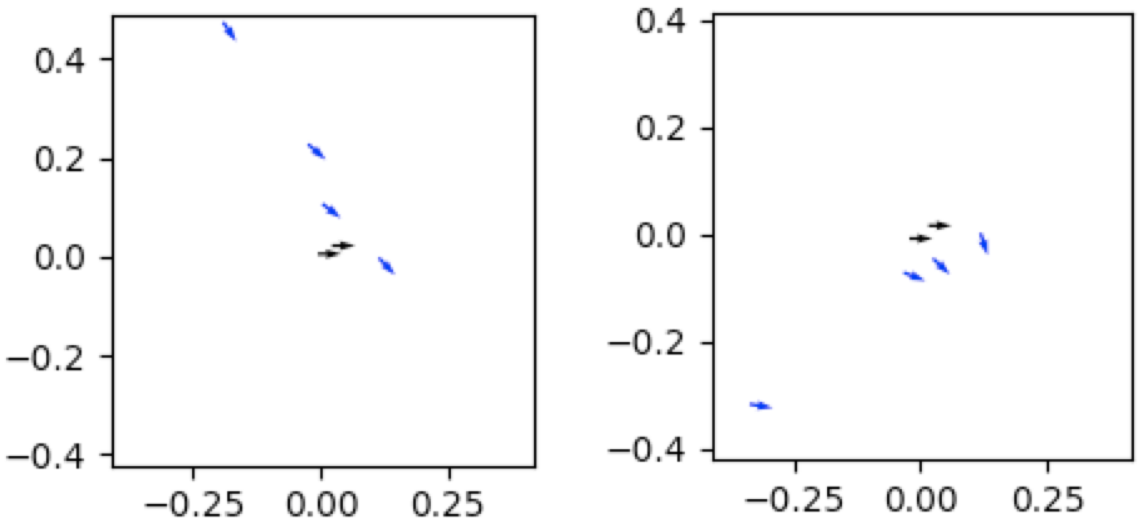
- An *unconstrained* point in the trajectory, and thus should be omitted from the tool transform model.
- An *orientation constraint*, where the rotation of the tooltip (and thus the end effector) is constrained more than its position (e.g. hooking a box is constrained more by the orientation of the hook than its position, as in Fig 5.3a).
- A *tooltip constraint*, where the position of the tooltip is constrained more than its rotation (e.g. sweeping a surface with a brush). Note that the *tooltip* position is the center of this constraint rather than the end-effector itself, and thus the range of valid

end-effector positions forms an arc around the tooltip, and its orientation remains angled toward the tooltip (e.g. Fig 5.3b).

We define two *tool transform models*, each reflecting either orientation or tooltip constraints. We fit the corrections matrix to each tool transform model, using RANSAC [89] to iteratively estimate the parameters of each model while discarding outlier and unconstrained correction data points. Each iteration involves (i) fitting parameter values to a sample of n datapoints, (ii) identifying a set of inlier points that also fit those model parameters within an error bound of ϵ, and (iii) storing the parameter values if the inlier set represents a ratio of the dataset $> d$. The RANSAC algorithm relies on a method for fitting parameters to the sample data, and a distance metric for a datapoint based on the model parameters. These are not defined by the RANSAC algorithm, and so we specify the parameterization and distance metric according to the tool transform model used, which we describe more in the following sections. We define an additional method to convert the best-fitting parameters following RANSAC completion into a typical transform that can be applied to poses.

5.2.2 Linear Tool Transform Model

Based on the *orientation* constraint type, we first consider a linear model for correction data, where corrections fitting this model share a linear relationship between the translational components of the corrections, while maintaining a constant relationship between the rotational components of corrections (visualized in Figure 5.4a). We model this linear relationship as a series of coefficients obtained by applying PCA to reduce the 3D position corrections to a 1D space.



(a) (b)

Figure 5.4: Each plot represents one set of corrections for a task. The position of each arrow represents the change in $< x, y >$ position, and points in the direction of the change in orientation introduced by that correction. Orientation constraints can be seen in (a), where the majority of corrections on this tool have low variance in their orientation, but higher variance in their x-y position. Tooltip constraints can be seen in (b), where the majority of corrections arc around a singular center of rotation, and orientation is dependent on the x-y position. Unconstrained keyframes (colored grey) are located near (0,0).

RANSAC Algorithm Parameters

The RANSAC algorithm is performed for k iterations, where we use the estimation

$$k = \frac{log(1.0 - p)}{log(1.0 - w^n))} \tag{5.5}$$

with desired confidence $p = 0.99$ and estimated inlier ratio $w = 0.5$. Additional parameters are as follows: $n = 2$ is the number of data points sampled at each RANSAC iteration, $\epsilon = 0.01$ is the error threshold used to determine whether a data point fits the model, and $d = 0.5$ is the minimum ratio between inlier and outlier data points in order for the model to be retained.

Model Parameter Fitting

Model fitting during each iteration of RANSAC consists of reducing the datapoints to a 1D model using PCA, returning the mean translational correction and the coefficients for the first principal component of the sample $\mathbf{S}$:

$$\Theta_{\text{linear}}(\mathbf{S}) = \langle \theta_\mu, \theta_\mathbf{u} \rangle \qquad \theta_\mu = \frac{1}{|\mathbf{S}|} \sum_{\mathbf{p} \in \mathbf{S}} \mathbf{p_t} \tag{5.6}$$

where $\mathbf{p_t}$ is the 3 x 1 translational difference indicated by the correction $\mathbf{p}$, $\mathbf{S}$ is the subset of the corrections matrix $\mathbf{C}$ sampled during one iteration of RANSAC such that $\mathbf{S} \subset \mathbf{C}$, and $\theta_\mathbf{u}$ is the eigenvector corresponding to the largest eigenvalue of the covariance matrix $\Sigma = \frac{1}{|\mathbf{S}|} \mathbf{S_t}^T \mathbf{S_t}$.

Error Function

Each iteration of RANSAC calculates the total error over all data points fitting that iteration's model parameters. We define the error of a single correction datapoint $\mathbf{p}$ as the sum of its reconstruction error and difference from the average orientation correction, given the

current model parameters θ:

$$\begin{aligned}\delta_{\text{linear}}(\mathbf{p},\theta) =& \|\mathbf{p_t} - (\theta_\mu + (\mathbf{p_t} - \theta_\mu)^T \theta_\mathbf{u} {\theta_\mathbf{u}}^{T+})\| \\ &+ \gamma\left((1 - \overline{\mathbf{q}}_\mathbf{n} {\mathbf{p_n}}^T)^2\right)\end{aligned} \tag{5.7}$$

where $\mathbf{x}^+$ indicates the Moore-Penrose pseudo-inverse of a vector, $\mathbf{p_n}$ is the unit vector representing the orientation difference indicated by the correction $\mathbf{p}$, $\overline{\mathbf{q}}_\mathbf{n}$ is a unit vector in the direction of the average rotation sampled from the model (defined in the next section), and γ is the weight assigned to rotational error ($\gamma = 1$ in our evaluations).

Sampling Function

After RANSAC returns the optimal model parameters and corresponding set of inlier points $\hat{\mathbf{I}} \subset \mathbf{C}$, the rotation and translation components of the transformation are sampled from the model. We define the sampling function according to the estimated "average" rotation $\overline{\mathbf{q}}$:

$$\Psi(\hat{\mathbf{I}}, \hat{\theta})_{\text{linear}} = \left\langle \overline{\mathbf{q}}, \overline{\mathbf{t}} \right\rangle \tag{5.8}$$

$$\overline{\mathbf{q}} = \underset{\mathbf{q}\in\mathbb{S}^3}{\arg\max}\, \mathbf{q}^T M \mathbf{q} \qquad M = \frac{1}{|\hat{\mathbf{I}}|} \sum_{\mathbf{p}\in\hat{\mathbf{I}}} {\mathbf{p_q}}^i {\mathbf{p_q}}^{iT} \tag{5.9}$$

The solution to $\overline{\mathbf{q}}$ for this maximization problem is the eigenvector corresponding to the largest eigenvalue of M [90]. The sample translation $\overline{\mathbf{t}}$ is the 3D offset corresponding to the mean value $\overline{z}$ from the 1D projection space:

$$\overline{\mathbf{t}} = \hat{\theta}_\mu + \overline{z} \hat{\theta_\mathbf{u}}^{T+} \qquad \overline{z} = \frac{1}{|\hat{\mathbf{I}}|} \sum_{p\in\hat{\mathbf{I}}} (\mathbf{p}_t - \hat{\theta}_\mu)^T \hat{\theta}_\mathbf{u} \tag{5.10}$$

5.2.3 Rotational Tool Transform Model

We now consider a model for corrections reflecting a *tooltip constraint*, in which we make the assumption that corrections indicate a constraint over the tool tip's *position*. Since the tool tip is offset from the end-effector, the position and rotation of the end-effector are constrained by each other such that the end-effector revolves around the tool tip (visualized in Figure 5.4b). We model this relationship by identifying a center-of-rotation (and corresponding rotation radius) for the tool tip, from which we can sample a valid end-effector position and rotation.

RANSAC Algorithm Parameters

We use the same parameters for k, w, d as in the linear model. We sample $n = 3$ points at each iteration, and use the error threshold $\epsilon = 0.25$. We define functions for model parameterization, error metrics, sampling, and variance in the following sections.

Model Parameter Fitting

We define the optimal model parameters for each iteration of RANSAC as the center-of-rotation (and corresponding rotation radius) of that iteration's samples $\mathbf{S}$:

$$\Theta_{\text{rotation}}(\mathbf{S}) = \langle \theta_{\mathbf{c}}, \theta_r \rangle \tag{5.11}$$

where $\theta_{\mathbf{c}}$ is the position of the center-of-rotation that minimizes its distance from the intersection of lines produced from the position and orientation of each correction sample:

$$\theta_{\mathbf{c}} = \arg\min_{\mathbf{c}} \sum_{i=1}^{|S|} D(\mathbf{c}; \mathbf{a_i}, \mathbf{n_i})^2 \tag{5.12}$$

where $\mathbf{a_i}$ and $\mathbf{n_i}$ are the position and unit direction vectors, respectively, for sample i in $\mathbf{S}$:

$$\mathbf{a}_i = [x_i, y_i, z_i]^T \qquad \mathbf{n}_i = \left(\mathbf{q}_i \cdot [0, 1, 0, 0]^T)\right) \cdot \mathbf{q}' \tag{5.13}$$

Here, $\mathbf{q_1} \cdot \mathbf{q_2}$ refers to the Hamilton product between two quaternions, and $\mathbf{q}'$ is the inverse of the quaternion $\mathbf{q}$:

$$\mathbf{q}' = [w, x, y, z]'^T = [w, -x, -y, -z]^T \tag{5.14}$$

We solve for the center-of-rotation by adapting a method for identifying the least-squares intersection of lines [91]. We consider each sample i to be a ray originating at the point $\mathbf{a_i}$ and pointing in the direction of $\mathbf{n_i}$. The center-of-rotation of a set of these rays is thus the point that minimizes the distance between itself and each ray. We define this distance as the piecewise function:

$$D(\mathbf{c}; \mathbf{a}, \mathbf{n}) = \begin{cases} \|(\mathbf{c} - \mathbf{a}) - d \cdot \mathbf{n}\|_2 & \text{if } d > 0 \\ \|\mathbf{c} - \mathbf{a}\|_2 & \text{otherwise} \end{cases} \tag{5.15}$$

where d is the distance between $\mathbf{a}$ and the projection of the candidate centerpoint $\mathbf{c}$ on the ray:

$$d = (\mathbf{c} - \mathbf{a})^T \mathbf{n} \tag{5.16}$$

We solve for $\theta_\mathbf{c}$ using the SciPy implementation of the Levenberg-Marquardt method for non-linear least-squares optimization, supplying Equation 5.15 as the cost function. We then solve for the radius corresponding to $\theta_\mathbf{c}$:

$$\theta_r = \frac{1}{|\mathbf{S}|} \sum_{i=0}^{|S|} \|\mathbf{a_i} - \theta_\mathbf{c}\| \tag{5.17}$$

Error Function

We define the error of a single data point $\mathbf{p}$ as its distance from the current iteration's center-of-rotation estimate:

$$\delta_{\text{rotation}}(\mathbf{p}, \theta) = \left(\frac{D(\mathbf{c}; \mathbf{a_p}, \mathbf{n_p})}{d_p} \right)^2 \tag{5.18}$$

Where d_p is defined in Equation 5.16.

Sampling Function

After RANSAC returns the optimal model parameters and corresponding set of inlier points $\hat{\mathbf{I}} \subset \mathbf{C}$, the rotation component of the transformation is first sampled using the "average" rotation $\overline{\mathbf{q}}_c$ from $\hat{\theta_\mathbf{c}}$ to all inlier points:

$$\overline{\mathbf{q}}_c = \underset{\mathbf{q} \in \mathbb{S}^3}{\arg\max}\, \mathbf{q}^T M \mathbf{q} \qquad M = \frac{1}{|\hat{\mathbf{I}}|} \sum_{\mathbf{p} \in \hat{\mathbf{I}}} \mathbf{r_p} \mathbf{r_p}^T \tag{5.19}$$

Where $\mathbf{r_p}$ is the quaternion rotation between $\hat{\theta}_\mathbf{c}$ and the position of $\mathbf{p}$, defined by normalizing the quaternion consisting of the scalar and vector parts:

$$\mathbf{r_p} = \left\langle \|\mathbf{a}\|^2 + \mathbf{b}\mathbf{a}^T, \mathbf{b}^T \times \mathbf{a} \right\rangle \tag{5.20}$$

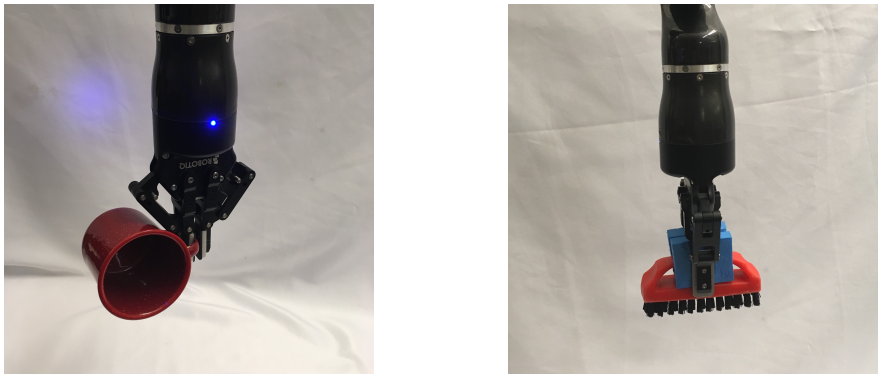
$$\mathbf{a} = \mathbf{p}_t - \hat{\theta}_\mathbf{c} \qquad \mathbf{b} = [\|\mathbf{a}\|, 0, 0] \tag{5.21}$$

The optimal $\overline{\mathbf{q}}_\mathbf{c}$ is the eigenvector corresponding to the largest eigenvalue of M; this represents the sampled rotation from $\hat{\theta}_\mathbf{c}$.

We then sample $\overline{\mathbf{t}}$ by projecting the point at distance $\hat{\theta}_r$ from $\hat{\theta}_\mathbf{c}$ in the direction of $\overline{\mathbf{q}}_\mathbf{c}$:

$$\overline{\mathbf{t}} = \hat{\theta}_\mathbf{c} + \left[(\overline{\mathbf{q}}_\mathbf{c} \cdot [0, \hat{\theta}_r, 0, 0]^T) \cdot \overline{\mathbf{q}}_\mathbf{c}{}' \right]_{1..3} \tag{5.22}$$

(a) (b) (c)



(d) (e)

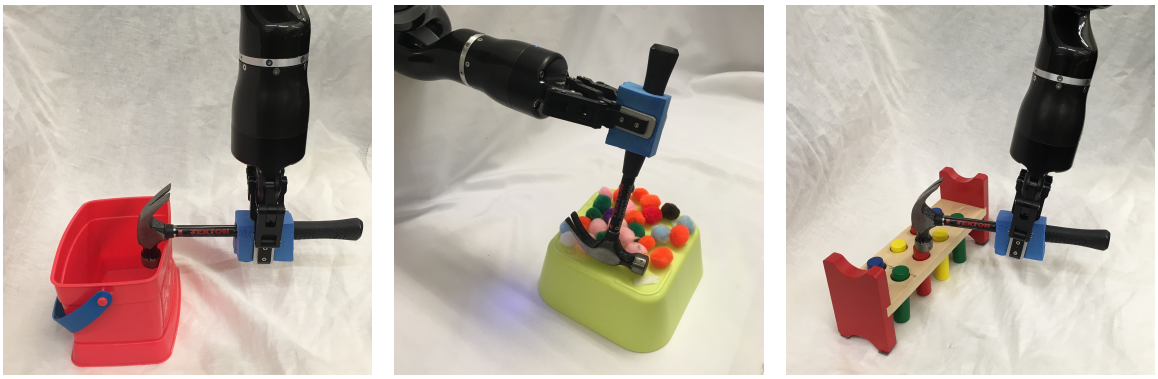
Figure 5.5: Tools a-c were used to demonstrate the three tasks shown in Figure 5.6, later transferred to use tools d-e. These tools exhibit a wide range of grasps, orientations, dimensions, and tooltip surfaces.

Where $\mathbf{x}_{1..3}$ indicates the 3 x 1 vector obtained by ommitting the first element of a 4 x 1 vector $\mathbf{x}$. Finally, we return the sample consisting of the translation $\overline{\mathbf{t}}$ and the normalized rotation $\overline{\mathbf{q}}$ between $\overline{\mathbf{t}}$ and $\hat{\theta}_{\mathbf{c}}$:

$$\Psi(\hat{\mathbf{I}}, \hat{\theta})_{\text{rotation}} = \left\langle \frac{\overline{\mathbf{q}}}{\|\overline{\mathbf{q}}\|}, \overline{\mathbf{t}} \right\rangle \tag{5.23}$$

$$\overline{\mathbf{q}} = \left\langle \hat{\theta}_r \|\mathbf{a}\| + \mathbf{b}\mathbf{a}^T, \mathbf{b}^T \times \mathbf{a} \right\rangle \tag{5.24}$$

$$\mathbf{a} = \hat{\theta}_{\mathbf{c}} - \overline{\mathbf{t}} \qquad \mathbf{b} = [\hat{\theta}_r, 0, 0] \tag{5.25}$$



(a) (b) (c)

Figure 5.6: (a) Hooking task, (b) sweeping task, and (c) hammering task

5.2.4 Best-Fit Model Selection

The linear and rotational tool transform models represent two different relationships between the translational and rotational components of corrections. We now define a metric for selecting between these two models based on how well they fit the correction data:

$$\Psi(\mathbf{C})_{\text{best-fit}} = \begin{cases} \Psi(\hat{\mathbf{I}}_l, \hat{\theta}_l)_{linear} & \text{if } \Delta_{linear} < \Delta_{rotation} \\ \Psi(\hat{\mathbf{I}}_r, \hat{\theta}_r)_{\text{rotation}} & \text{otherwise} \end{cases} \tag{5.26}$$

Where $\hat{\mathbf{I}}_l, \hat{\theta}_l, \hat{\mathbf{I}}_r, \hat{\theta}_r$ represent the optimal inlier points and parameter values from the linear and rotational models, respectively. The fit of the linear model is calculated as its range of values $\mathbf{z}$ projected in the model's 1D space:

$$\Delta_{linear} = \text{range}(\mathbf{z}) \qquad \mathbf{z} = \{(\mathbf{p}_t - \hat{\theta}_\mu)^T \hat{\theta}_\mathbf{u} | \forall \mathbf{p} \in \hat{\mathbf{I}}\} \tag{5.27}$$

The fit of the rotational model is calculated as the range of unit vectors in the direction of each inlier point as measured from the center-of-rotation:

$$\Delta_{rotation} = 1 - \frac{1}{|\hat{\mathbf{I}}|} \left\| \sum_{p \in \hat{\mathbf{I}}} \left[(\mathbf{r_p} \cdot [0,1,0,0]^T) \cdot \mathbf{r_p}' \right]_{1..3} \right\|_2 \tag{5.28}$$

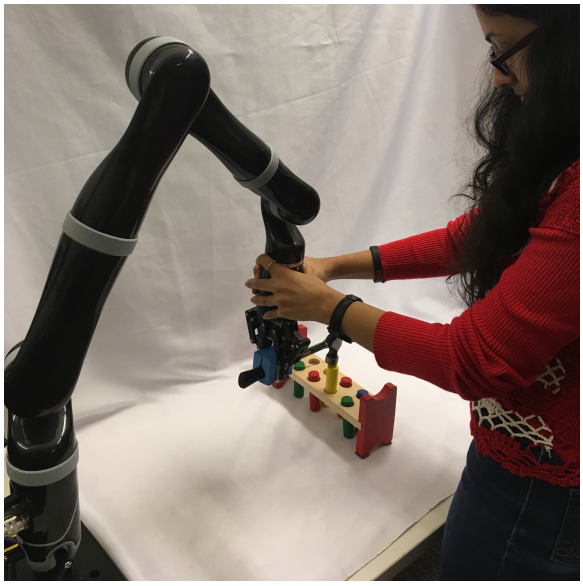


Figure 5.7: The robot receiving a demonstration of a hammering task

where $\mathbf{r_p}$ is defined in Equation 5.20.

5.3 Evaluating Transfer by Correction

We evaluated the transfer by correction algorithm results on a 7-DOF Jaco2 arm equipped with a Robotiq 85 gripper and mounted vertically on a table-top surface (pictured in Figure 5.7). Each evaluation configuration consisted of: (i) one task demonstration provided using the *source* tool, (ii) the new, *replacement* tool, and (iii) one correction task (demonstrated with the source tool, and used to obtain corrections with the replacement tool). We describe data collection for each of these steps in the following sections.

5.3.1 Collecting Task Demonstrations

Three tasks (Figure 5.6) were demonstrated using three prototypical, "source" tools (Figure 5.5a-c), resulting in a total of 9 demonstrations. Demonstrations began with the arm positioned in an initial configuration, and with the gripper already grasping the tool. Objects on the robot's workspace were reset to the same initial position before every demonstration. We provided demonstrations by indicating keyframes [13] along the trajectory,

each of which was reached by moving the robot's arm to the intermediate pose. At each keyframe, the 7D end effector pose was recorded; note that this is the pose of the joint holding the tool, and *not* the pose of the tool-tip itself (since the tool-tip is unknown to the robot). We provided one keyframe demonstration for each combination of tasks and source tools in this manner, each demonstration consisting of 7-12 keyframes (depending on the source tool used) for the sweeping task, 10-11 keyframes (depending on the source tool used) for the hooking task, and 7 keyframes for the hammering task. Following each demonstration, a Dynamic Movement Primitive (DMP) model [25, 27] was trained on the recorded keyframe trajectory. DMPs represent a demonstration as a stable dynamical system and are generalizable to variations in start and end pose constraints. We re-recorded the demonstration if the trained DMP failed to repeat the demonstration task with the source tool.

5.3.2 Recording Interactive Corrections

Following training, the arm was reset to its initial configuration, with the gripper already grasping a new tool (Figure 5.5d-e). Objects on the robot's workspace were reset to the same initial position as in the demonstrations. The learned model was then used to plan a trajectory in task-space, which was then converted into a joint-space trajectory using TracIK [92] and executed, pausing at intervals defined by the keyframe timing used in the original demonstration. When execution was paused, it remained paused until the arm pose was confirmed. If no correction was necessary, the pose was confirmed immediately; otherwise, the arm pose was first corrected by moving the arm to the correct position.

Two poses were recorded for each correction: (i) the original end-effector pose the arm attempted to reach (regardless of whether the goal pose was reachable with the new tool), and (ii) the end-effector pose following confirmation (regardless of whether a correction was given). Trajectory execution then resumed from the arm's current pose, following the original task-space trajectory so that pose corrections were not propagated to the rest of

the trajectory. This process continued until all keyframes were corrected and executed, resulting in the correction matrix $\mathbf{C}$ (Equation 5.2).

5.3.3 Performance Measures

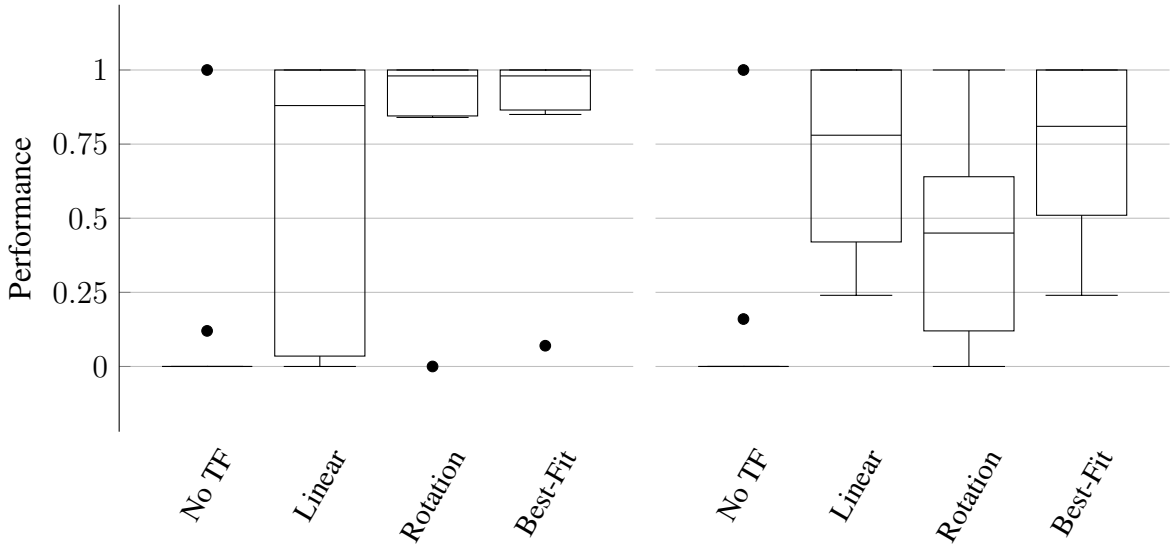
For each transfer execution, we measured performance according to a metric specific to the task:

- *Sweeping:* The number of pom-poms swept off the surface of the yellow box.
- *Hooking:* The final distance between the box's target position and the closest edge of the box (measured in centimeters).
- *Hammering:* A binary metric of whether the peg was pressed any lower from its initial position.

5.3.4 Within-task Transfer Results

Within-task performance measures the algorithm's ability to model the corrections and perform the corrected task successfully. Transfer was performed using the transform model learned from corrections on *that same task/tool combination*. For example, for the sweeping task model learned using the hammer, corrections were provided on the replacement tool (e.g. a mug) and then used to perform the sweeping task using that same mug. For each source tool, we evaluated performance on all 3 tasks using each of the 2 replacement objects, resulting in 18 sets of corrections (one for each combination of task, source tool, and replacement tool) per tool transform model (linear and rotational).

Using the better-performing model resulted in $\geq 85\%$ of maximum task performance in 83% of cases. The better-performing model was selected using the best-fit metric in 72% of cases. Figure 5.9 lists the percentage of transfer executions (using the best-fit model) that achieve multiple performance thresholds, where best-fit results were recorded as the performance of the model returned by Equation 5.26.

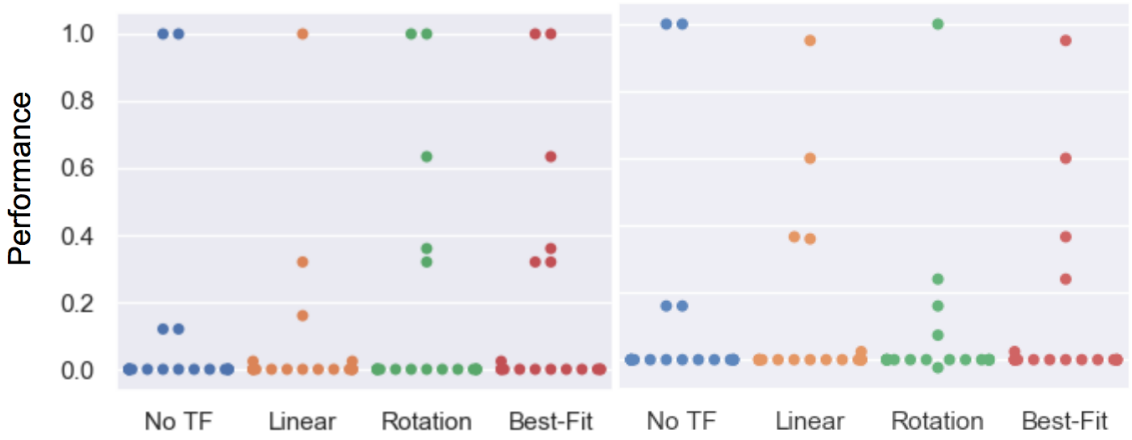


(a) Brush (b) Mug

Figure 5.8: Results for within-task transfer using the scrub-brush or mug as the replacement tool. Performance was measured according to the metrics in Section 5.3.3, scaled between 0-1.

Performance Threshold	**95%**	**85%**	**75%**	**65%**	**55%**	**45%**
Transfer Executions	61%	72%	72%	72%	83%	89%
Untransformed Executions	11%	11%	11%	11%	11%	11%

Figure 5.9: Percentage of within-task transfer executions (selected by best-fit model) and untransformed trajectories achieving various performance thresholds (defined as the % of maximum performance metric for that task, described in Section 5.3.3)



(a) Brush (b) Mug

Figure 5.10: Results for across-task transfer using the scrub-brush or mug as the replacement tool. Performance was measured according to the metrics in Section 5.3.3, scaled between 0-1.

We scaled the result of each transfer execution between 0 and 1, with 0 representing the initial state of the task and 1 representing maximum performance according to the metrics in Section 5.3.3. Figure 5.8 reports the performance distribution aggregated over all tasks, transferred from each of the 3 source tools to either the scrub-brush (pictured in Figure 5.5e, results in Figure 5.8a) or mug (pictured in Figure 5.5d, results in Figure 5.8b) as the replacement tool. The mean performance results are reported in Figure 5.11a, with darker cells indicating better performance. Overall, the transform returned using the best-fit metric resulted in average performance of 6.9x and 5.9x that of the untransformed trajectory when using the scrub-brush and mug, respectively, as replacement tools.

5.3.5 Across-task Transfer Results

Across-task transfer performance measures the generalizability of corrections learned on one task when applied to a *different* task using the same tool, without having received any corrections on that tool/task combination. For example, the hooking task was learned using the hammer, and transferred to the mug using corrections obtained on the sweeping task. We evaluated 36 total transfer executions (one per combination of demonstration task,

Replacement Tool	No TF	Linear	Rotation	Best-fit
Brush	0.124	0.648	0.850	**0.863**
Mug	0.129	0.738	0.489	**0.765**

(a) Mean performance of within-task transfer to the brush and mug replacement tools over all 18 transfer executions for each tool.

Replacement Tool	No TF	Linear	Rotation	Best-fit
Brush	0.024	0.053	0.268	**0.302**
Mug	0.097	0.162	0.101	**0.162**

(b) Mean performance of across-task transfer to the brush and mug replacement tools over all 18 transfer executions for each tool.

Replacement Tool	No TF	Linear	Rotation	Best-fit
Brush	0.024	0.053	0.268	**0.302**
Mug	0.097	0.162	0.101	**0.162**

(c) Mean performance of across-task transfer to the brush and mug replacement tools over the subset of transfer executions in which the transformation between source and correction tasks is similar for the source and replacement tool (10 executions for the brush, 12 for the mug).

Figure 5.11: Mean performance for within-task, across-task, and a subset of across-task transfer executions. Darker cells indicate higher average performance.

source tool, correction task (distinct from the demonstration task), and replacement tool) per tool transform model (linear and rotational).

Figure 5.10 reports the performance distribution aggregated over all tasks, transferred from each of the 3 source tools to either the scrub-brush (Figure 5.10a) or mug (Figure 5.10b) as the replacement tool. The mean performance results are reported in Figure 5.11b, with darker cells indicating better performance. Overall, the transform returned using the best-fit metric resulted in average performance of 1.6x and 0.94x that of the untransformed trajectory when using the scrub-brush and mug, respectively, as replacement tools. The performance distribution is improved when using the transform learned from corrections, resulting in 2.25x as many task executions achieving $\geq 25\%$ of maximum task performance.

In order to understand the conditions under which a transform can be reused successfully in the context of another task, we also report the mean performance results for a subset of the across-task executions (Figure 5.11c). This subset consists of only the task executions where the relative orientation is the same between (i) the source tool's tooltips used for the source and target tasks and (ii) the replacement tool's tooltips used for the same two tasks. This subset consisted of 10 executions for the scrub-brush, and 12 for the mug. Overall, for this subset of executions, the transform returned using the best-fit metric resulted in average performance of 12.6x and 1.7x that of the untransformed trajectory when using the scrub-brush and mug, respectively, as replacement tools.

5.4 Implications of Within- and Across-Task Transfer Results

Our within-task transfer evaluation tested whether we can model the transform between two tools in the context of the same task (represented by the solid blue arrow in Figure 5.12) using corrections. Our results indicate that one round of corrections is sufficient to indicate this relationship between tools; collectively, the linear and rotational models achieved $\geq$ 85% of maximum task performance in 83% of cases. Individually, the models selected

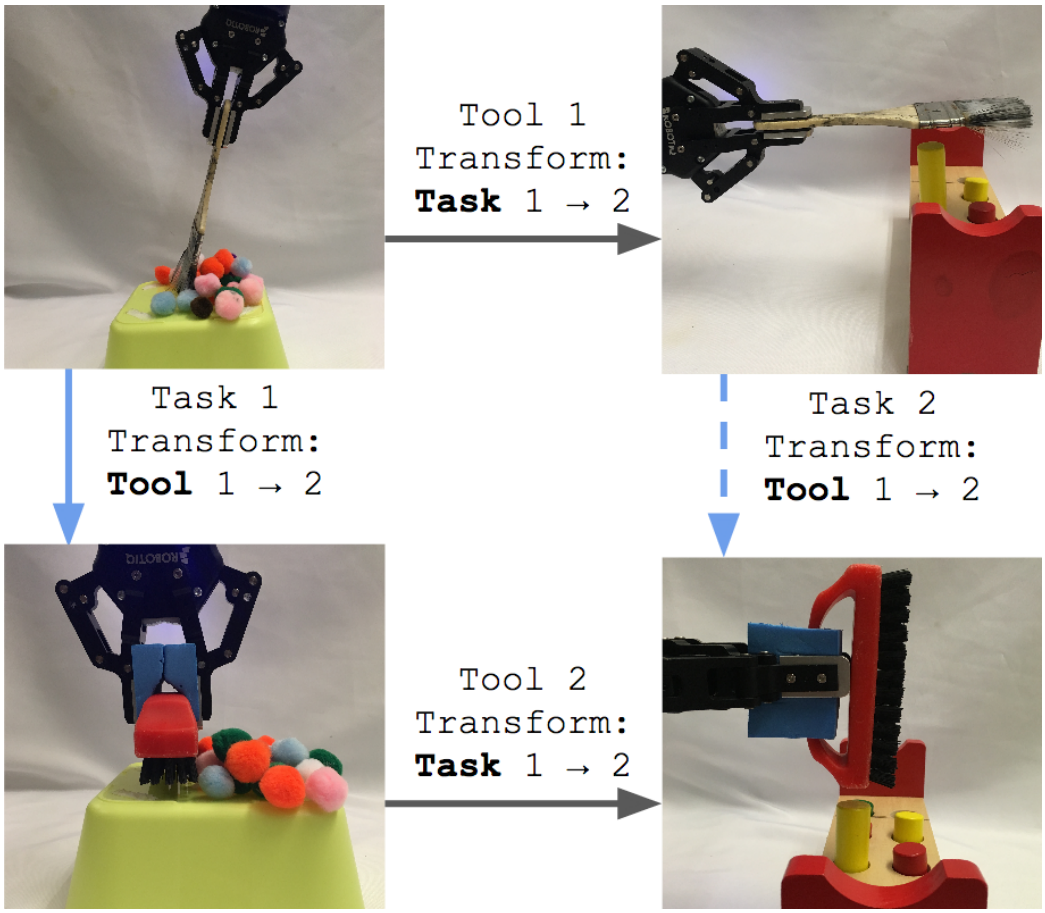


Figure 5.12: Corrections indicate the transform from tool 1 to tool 2 for the same task (indicated by the solid blue arrow). Our within-task transfer evaluation tested whether we can use corrections to sufficiently model this relationship. Different tasks may use different tooltips from the same tool (such as the different tooltips used to complete tasks 1 and 2). Our across-task evaluation tests whether the transform learned from corrections (solid blue arrow) can be reused as the transform between the two tools for another task (indicated by the dashed blue arrow).

by the best-fit metric achieved this performance threshold in 72% of cases. This indicates that, in general, the fit of the model itself can be used to indicate the relationship between end-effector position and orientation for a given tool/task combination.

Aside from analyzing high task performance, we are also interested in whether our approach enables graceful degradation; even if the robot is unable to complete the task fully with a new tool, ideally it will still have learned a transform that enables *partial* completion of the task. The results shown in Figure 5.9 demonstrate that Transfer by Correction offers robust behavior such that even when it results in sub-optimal performance, it still meets lower performance thresholds in nearly 90% of cases. In contrast, the untransformed baseline does not meet lower performance thresholds, and thus produces all-or-nothing results that lack robustness.

The primary benefit of modeling corrections (as opposed to re-learning the task for the new tool) is two-fold: First, the robot learns a transformation that reflects *how* the task has changed in response to the new tool, which is potentially generalizable to other tasks (as we discuss next). We hypothesize that in future work, this learned transform could be parameterized by features of the tool (after corrections on multiple tools). Second, since we do not change the underlying task model, but instead apply the learned transform to the resulting trajectory, the underlying task model is left unchanged. We expect that this efficiency benefit would be most evident when transferring a more complex task model trained over many demonstrations; rather than require more demonstrations with the new tool in order to re-train the task model, the transform would be applied to the result of the already-trained model.

We have also explored how well this transform generalizes to other tasks. Different tooltips on the same tool may be used to achieve different tasks, such as how the end and base of the paintbrush are used to perform sweeping and hammering tasks, respectively, in Figure 5.12. While we do not explicitly model the relationship between tooltips on the same tool (represented by the top grey arrow in Figure 5.12), they are inherent to the

learned task models. A similar relationship exists for the replacement tool (represented by the bottom grey arrow in Figure 5.12). Our across-task evaluation seeks to answer whether the relationship between tools in the context of the first task (solid blue arrow) can be reused for a second task (represented by the dashed blue arrow) without having received any corrections on that tool/task combination (tool 2 and task 2). While we see lower performance in across-task evaluations compared to the within-task evaluations, it does improve transfer in 27.8% of across-task transfer executions (in comparison to the untransformed trajectory).

In the general case, our results also indicate that we cannot necessarily reuse the learned transformation on additional tasks, as average performance in across-task transfer is slightly worse than that of the untransformed trajectory when the mug is used as a replacement tool. This presents the question: given a transform between two tools in the context of one task, under what conditions can that transform be reused in the context of another task *without additional corrections or training*? We do see that across-task performance is greatest when considering only the subset of cases where the relationship between the tooltips used in either task is similar for the source and replacement tools (in our evaluation, this is 10 of 18 executions using the brush, and 12 of 18 executions using the mug). Within this subset, across-task transfer improves performance in 41% of transfer executions. From this we draw two conclusions: (i) the transform applied to a tool is contextually dependent on the source task, target task, and tooltips of the source and replacement tool, and (ii) a transform can be reused when the relationship between tooltips used in either task is similar for the source and replacement tools.

5.5 Summary

We have presented Transfer by Correction: a method of modeling a human teacher's corrections of the robot's motion when using a new tool. We have contributed two models for representing corrections, a linear and rotation model, that each represent a different rela-

tionship between the end-effector's position and orientation when using a tool. We have also presented a metric for choosing the better-fitting model for a set of corrections.

In our within-task evaluation, we have demonstrated that either the linear or rotational model is sufficient to represent corrections for successful task completion with the new tool in 83% of task executions. Furthermore, using a metric to select the best-fitting model resulted in improved performance in 89% of tasks (in comparison to the original, untransformed trajectory).

Our across-task evaluation tests the generalizability of the transforms learned from corrections to additional tasks, *without any additional training or corrections*. We observed that across-task transfer improved performance in 27.8% of task executions, and that further improvement is seen in transfer scenarios where the relationship between the tooltips used on the source tool is similar to that of the replacement tool. Overall, these results (published in [88]) indicate that successful task adaptation for a new tool is dependent on the the tool's usage within that task, and that the transform model learned from interactive corrections can be generalized to other tasks providing a similar context for the new tool.

5.5.1 Key Contributions and Insights

This chapter presents the first interactive method for modeling tool replacements. Our evaluation resulted in the following key findings:

Insight #1: Corrections provide a sample of the *constrained* transform between the tooltip and the robot's end-effector. This underlying constraint is task-dependent; our best-fit model results indicate that **multiple constraint types should be modeled and evaluated for each task**, with the best-fitting model used to produce the final transform output.

Insight #2: While the tooltip transform is task-specific, it can be applied to additional tasks under certain conditions. This is dependent on a second transform: the transform between *multiple tooltips* on the same tool. **A tooltip transform can be reused for an additional task when the transform between the tooltips used to complete (i) the cor-**

rected task and (ii) the additional task are similar for the source and target tools.

5.5.2 Open Questions

In this chapter, we have presented a corrections-based approach to sampling and modeling the transform resulting from a tool replacement. In doing so, we model a single, *static* transform for a particular tool/task pairing. We have evaluated how well this model transfers to other tasks using the same tool replacement. An extension of this work would consider transfer across *tools*.

We envision that a robot could not only model the transform samples obtained by interactive corrections, but also learn to generalize that model to other, similar tools. For example, after receiving corrections for one ladle for a scooping task, the robot would ideally be able to model those corrections such that it would apply to ladles of different shapes or proportions as well. Just as how this chapter discussed the underlying constraints that are sampled via corrections, we anticipate that a robot could learn an underlying relationship between object features (such as dimensions or concavity) and the resulting transform for that tool.

CHAPTER 6
META-LEARNING FOR TASK PARAMETERIZATION

Throughout this dissertation, we have analyzed the information needed for a robot to ground its abstracted task model in a new environment, and how that information can be obtained via interaction. In the previous chapter, we considered *across-task transfer*, in which a robot receives corrections when repeating one task using a new tool and, at a later time, reuses the corrections model to transfer other tasks for execution using the same new tool. We now consider how a robot can generalize across *tools* rather than across tasks; that is, the robot receives corrections for a set of tools and then extrapolates the corrections model so that the task can be transferred to a new, uncorrected tool. For example, the first column of tools shown in Figure 6.2 represents three classes of tools (hammers, spoons, and knives). The next column represents how each class of tools is instantiated by various objects (e.g. scoops with varied shapes and dimensions). The third column illustrates how each tool instance is further represented as a set of images of that tool taken from multiple viewpoints. We aim for the robot to learn a tool transform model that is generalizable across a category of tools (e.g. transfer within the second column of tool images in Figure 6.2).

In this chapter, we formalize this transfer problem and characterize the data required to address it. While we have found that the data needed to address this category of transfer problems is not yet available, we discuss what algorithms and datasets will enable future work to solve this category of problems.

6.1 Background: Adapting Meta-Learning for Across-Tool Transfer

Existing approaches for transfer typically require additional exploration with the new environment, or make assumptions about the relationship between the tool and its effect on task execution. While simulation provides one method of exploring this relationship, it requires

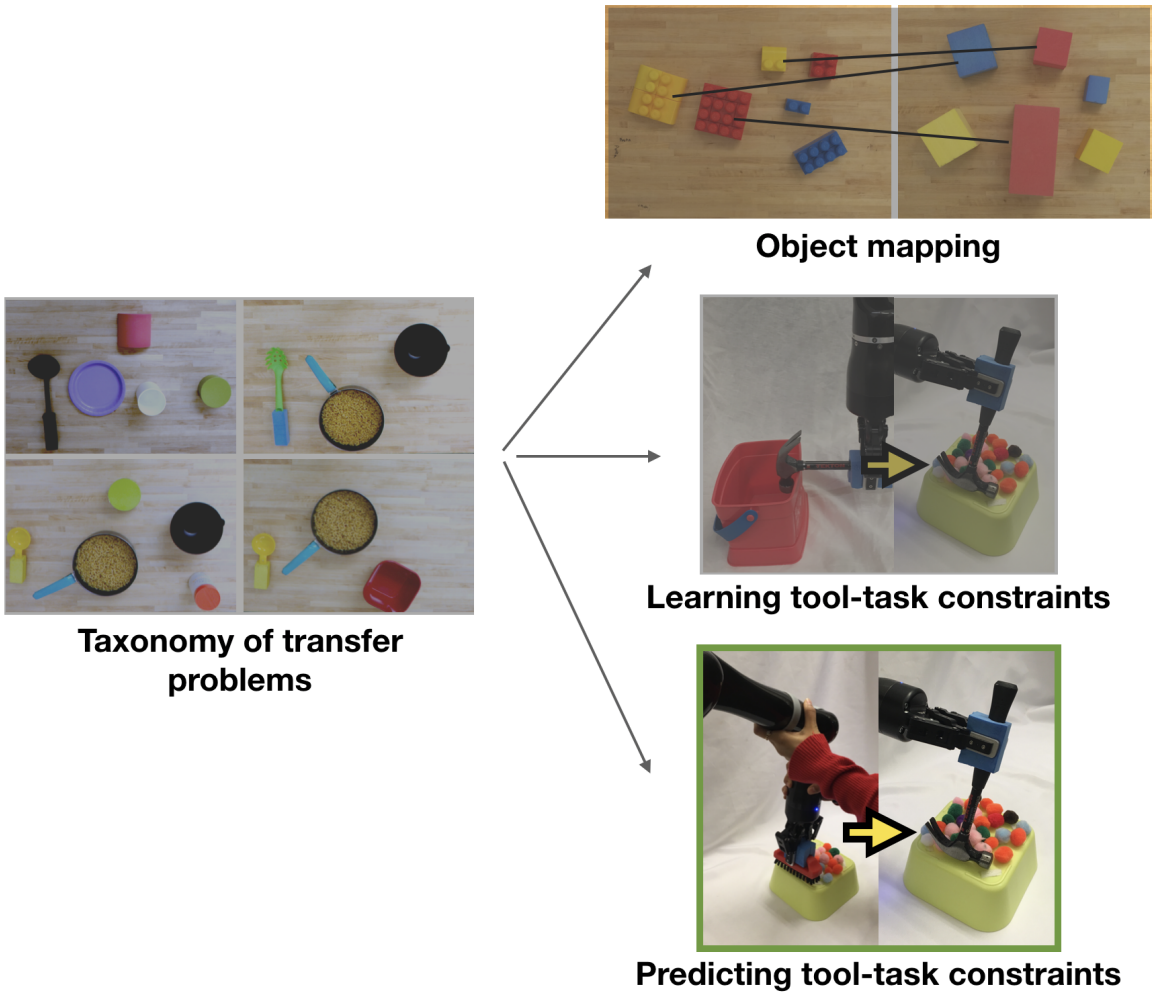


Figure 6.1: We now focus on a third category of transfer problems, highlighted in green: those in which an object replacement affects the robot's trajectory *and* can be parameterized according to visual features of the tool.

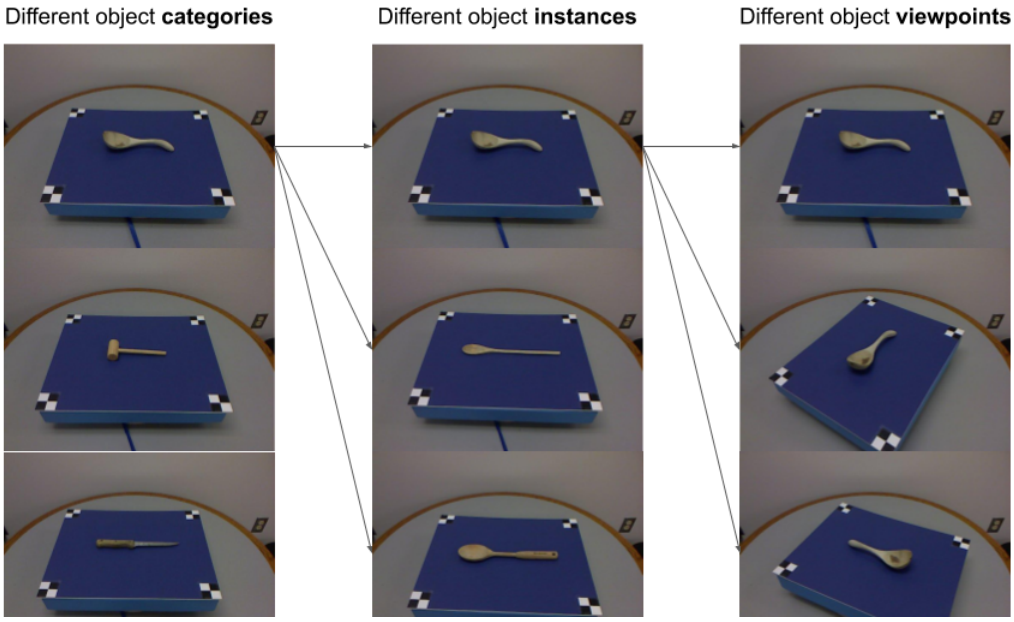


Figure 6.2: Tool categorization: Column 1 represents a set of tool categories. Column 2 represents a set of tool instances within a single category. Column 3 represents a set of images corresponding to a single tool instance. Images are from the UMD Part Affordance Dataset [93].

a known goal or reward function for the task being simulated, and may not generalize across tasks. Recent work in one-shot and meta-learning (described further in Chapter 2) enables an agent to adapt quickly to new variations of a task by performing background training over a large set of tasks and their variations. In doing so, this approach aims to directly learn a latent-space parameterization of a task that is easily tuned for a new environment using few demonstrations. At a later time, when a new task is presented, the agent only needs a limited number of labeled training datapoints from the new task in order to tune its model to that task. This enables the agent to leverage its extensive training on previous tasks in order to bootstrap its learning of the new task.

Meta-learning has been successfully applied to learning problems in computer vision domains and fully-simulated reinforcement learning problems. When applied to the domain of tool transfer, meta-learning would ideally enable a robot to use extensive background training to learn the common relationships between visual features and tooltips that

are shared by tools within their respective categories (e.g. cups, knives, scoops). When presented with a novel category of tools, the robot would then only need demonstrations using a small number of tools within the new category in order to learn the relationship between visual features and tooltips within that category. After this "tuning" stage, the robot should then be able to extrapolate this relationship to predict the tooltips for the remaining, undemonstrated tools in that same category.

To perform meta-learning in this manner, however, the robot would need access to a large dataset of input-output pairings consisting of the tool image (input) and resulting tooltip pose (output) measured with respect to the robot's end-effector. However, as demonstrated in Chapter 5, tooltips are task-specific; within a single tool, the tooltip used to complete one task (e.g. the surface of a hammer used to hammer a nail) is not necessarily the same as the tooltip used to complete another task (e.g. the side of the hammer may be used to sweep objects off a surface, or the claw-end of the hammer may be used to remove a nail). As a result, a dataset containing a single, canonical tooltip for each tool would fail to capture the task-contextual nature of tool use.

Training datasets for affordance prediction, such as the UMD Part Affordance Dataset [93] are relevant to this challenge. Rather than annotate images with a single tooltip, this dataset highlights the regions of an image that support each of seven different affordances (e.g. cutting, grasping, containing). Figure 6.3 illustrates these labeled affordance regions. As a result of containing labeled affordance regions, this dataset has the potential to highlight candidate tooltip "areas" that are consistent both across viewpoints of each object and across multiple objects within the same category (e.g. saws, ladles, mugs). While this dataset is relevant to predicting tooltips, it does not address the problem of specifying the relevant tooltip for a *particular task*. For example, the full blade of a knife may be labeled as enabling the "cutting" affordance (as shown in the green region in Figure 6.4), even though a cutting task is likely to be performed with respect to only the edge of the blade. Furthermore, since affordance data is presented in the form of pixel-wise image labels, it

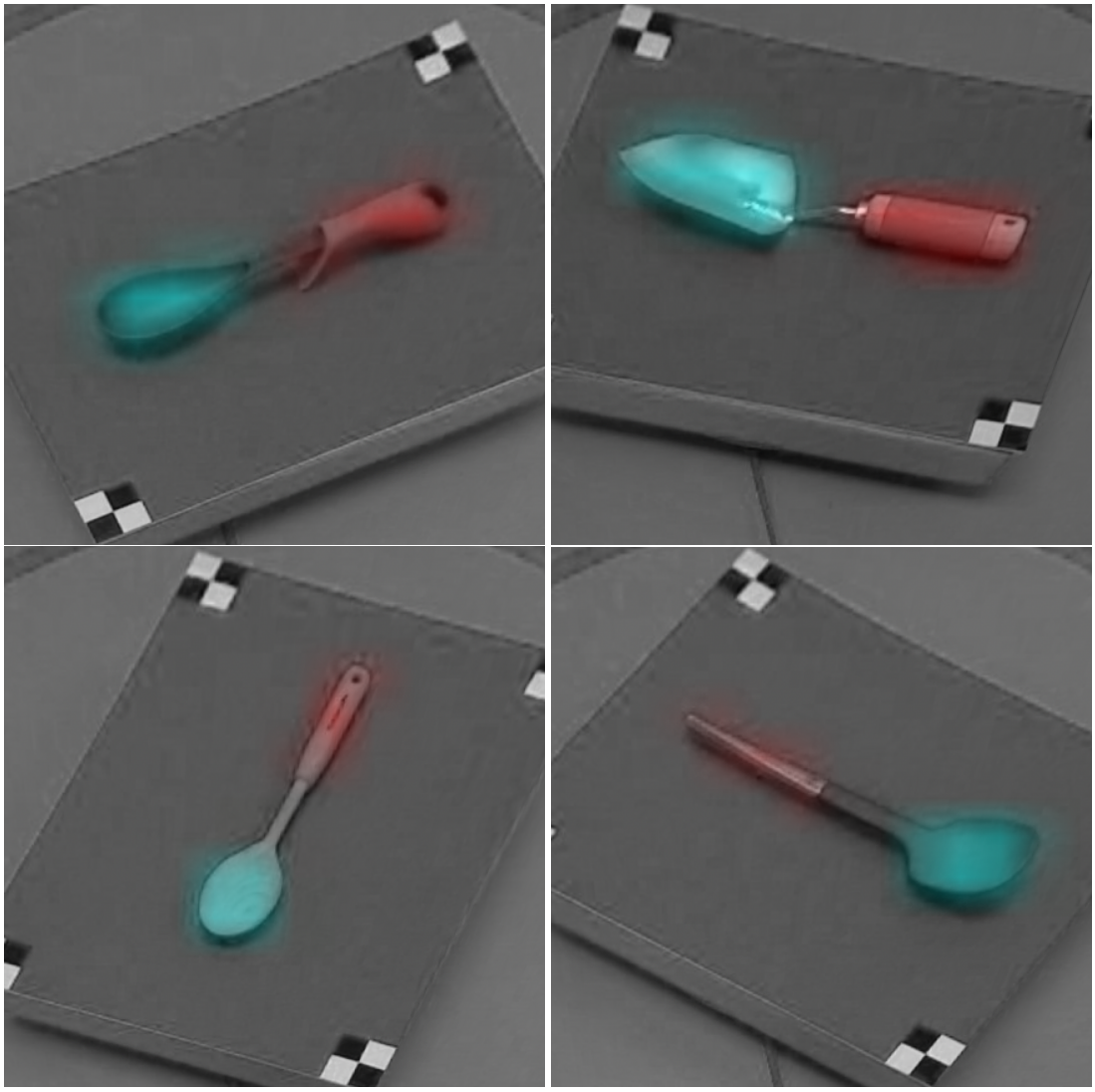


Figure 6.3: Various tools within the same "scoop" category. The blue region is derived from pixel-wise labels for the "scoop" affordance, and the red region is derived from labels for a "grasp" affordance [93]. These labels indicate the general area of potential tooltips, but do not indicate *which* point, edge, or surface is relevant for a specific task.

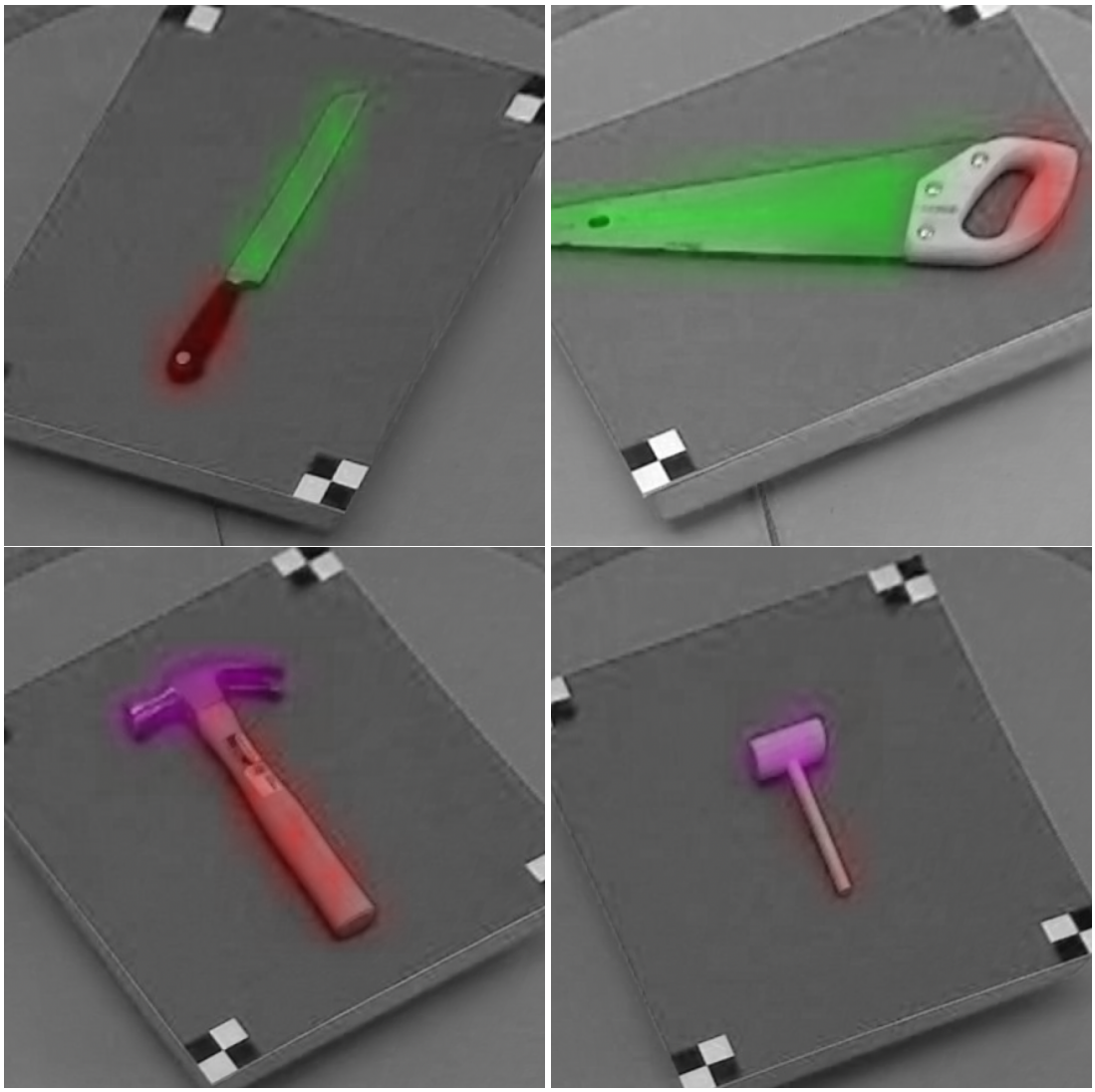


Figure 6.4: Affordance regions may be broad, spanning multiple possible tooltips. As a result, predicting the affordance region is not sufficient to plan with respect to that tool's tooltip. For example, the full blade surfaces of the saw and knife are labeled as enabling the "cutting" affordance (highlighted in green); however, cutting is only performed using the edge of the blade, and requires that the blade be oriented toward the cutting target. Similarly, different points of a hammer head may be enable different tasks (e.g. pounding versus prying), and thus detecting a task-independent affordance region (highlighted in purple) is not sufficient to plan a task trajectory.

does not provide any data concerning the kinematic implications of using this tool. Since the tool is observed and labeled from a static, overhead perspective, affordance data is only available along a single 2D plane, and thus does not indicate the *orientation* at which each affordance is or is not valid. This is essential for manipulating the tool properly; even if the robot were to determine that the tooltip of a knife is located along the edge of its blade, the blade must still be oriented carefully with respect to the cutting target for the task to be completed successfully. In summary, successful task completion relies on the robot having a model of the composite transform between (i) the end-effector, (ii) its grasp of the tool (highlighted in red in Figure 6.4), and (iii) the tooltip position and orientation.

Throughout this thesis, we have demonstrated a framework for addressing task-specific transfer problems; depending on the data needed to ground the task representation in a particular environment, the robot may use one of several interaction methods (e.g. gestures, demonstrations, or corrections) to query a human teacher for this data. We have accompanied each mode of interaction with an algorithm and representation used to model the data collected from interaction. We propose that these same principles can be applied to the problem of across-tool transfer; namely, that the robot can obtain and model task-specific tooltips from interaction.

In order to adapt meta-learning to this framework, we first consider the data required for background training and, later, tuning on the new tool category. We presume that visual input data can be obtained autonomously by the robot observing tools in its environment, and thus is readily available for multiple tools and at multiple perspectives. In contrast, demonstrations provide grounded tooltip data for a particular tool, but are sparsely available and can only be obtained one at a time for a specific tool-task pairing as requested by the robot on an as-needed basis. In this chapter, we propose (i) a neural network model that operates directly over depth images of the robot's environment and (ii) a training methodology for this approach that is tailored to the availability of different data types obtained by the robot. We propose leveraging two sources of information according to their availability:

(1) a large dataset of tool images, and (2) a limited number of pose transforms provided via demonstration.

6.2 Problem Specification

Tool-Task Groupings: Let O be the set of all tools, A be the set of all tasks, and T be the set of tool-task pairings such that for all $t \in T$, $t =< o, a >$ implies that task $a \in A$ can be completed using tool $o \in O$. A single tool may be used to complete multiple tasks, and so $|T| \geq |O|$. Suppose that there is a set of tool groupings G such that each grouping $g_i \in G$ contains the set of tools that may be used to complete task a_i:

$$g_i = \{o \in O | < o, a_i > \in T\}$$

Input/Output Representations: The robot may observe each tool $o \in O$ from multiple perspectives. Let I_o be the set of RGB-D images of the tool o. There are two forms of output whose values are specific to each tool-task pairing:

- $P_{o,a}$ contains the pixel-wise coordinate correlating to the tooltip in each image of tool o for the task a. Each $p^n_{o,a} \in P_o$ corresponds to image $i^n_{o,a} \in I_o$.
- $f_{o,a}$ is the transform between the robot's gripper pose and the tooltip for the tool-task pairing $< o, a >$. This transform is independent of the various image perspectives in I_o, and thus there is only one tooltip transform for the tool o in the context of the current task a.

Note that while the perceptual representation of tool remains the same regardless of the task being performed, the tooltip that is used to guide the robot's trajectory *is* dependent on the task (as discussed in Chapter 5). As a result, there is a one-to-many mapping between the visual representation of the tool and the transform or pixel-wise representation of the tooltip.

Training and Testing Datasets: We separate the tool groupings G into two sets: the offline training subset G_r and the online subset containing a tuning subset G_u and testing subset G_e. We define these sets as follows:

- The offline training set G_r consists of multiple tool groupings and contains data derived from offline image datasets. Thus, for each $< o, a >$ pair $g_i \in G_r$, the image set I_o and pixel-wise tooltip set $P_{o,a}$ are known. The robot-specific transform $f_{o,a}$ is not known.

- The tuning set G_u is fully-labeled and contains both forms of output $P_{o,a}$ and $f_{o,a}$ for the image set I_o. I_o and $P_{o,a}$ are derived from the robot's perception, and $f_{o,a}$ is derived from interaction (e.g. demonstrations or corrections) from the teacher.

- The testing set G_e contains unlabeled data; that is, only the image set I_o is derived from the robot's perception and thus is known for each tool $o \in G_e$, while both $P_{o,a}$ and $f_{o,a}$ are unlabeled.

Objective: We aim to identify a model $f = \phi(I_o)$ that accepts a set of tool images as input and produces a transform representation of the tooltip. The learning objective is to train ϕ over the training set G_r such that after being tuned on G_u it minimizes the mean squared error over the test set G_e.

6.3 Approach: Interactive Tool-Task Grounding

Meta-learning is a training methodology in which a model is trained over multiple tasks or task variations with the objective of minimizing error *after tuning* that model on a small training set for a new task. In doing so, meta-learning aims to (i) initialize the model parameters such that it bootstraps learning for a new task, and (ii) encode the common relationships between input/output pairs across different tasks. The primary benefit of meta-learning is that it enables a robot to leverage its prior training data collected over a variety

of tasks in order to quickly learn a new task using little training data.

We adapt meta-learning to predict the transform between the robot's gripper and the tooltip for each tool in a new category of tools (e.g. introducing a set of various hammers) and/or a new tool-task transform (e.g. learning to use the claw-end of the hammers). This prediction occurs after tuning the model over (i) images of object instances within the target class of tools and (ii) the transform for a particular tool instance within that class. Meta learning algorithms such as Model-Agnostic Meta-Learning (MAML) [38] could be directly used to train a non-linear regression model that predict this tooltip from an image of the tool. However, since multiple tasks may be performed using the same tool, and thus may use different tooltips of the same tool, there is a one-to-many relationship between tool images and their corresponding tooltips across multiple tasks. While meta learning enables model tuning for a new input type or a new function between input and output, but is not optimized to ground both simultaneously.

We propose *Interactive Tool-Task Grounding* (ITG): a meta-learning approach to quickly grounding the tooltip for a new tool and task. We consider the two sources of novelty that affect the expected model output: novel classes of tools, and novel relationship between the tool and tooltip. We account for both sources of novelty by adapting meta learning for a two-fold learning objective, considering both (i) weighting visual features according to their saliency in predicting the tooltip, and (ii) regression of the tooltip transform from the weighted feature locations. By incorporating both learning goals, we can take advantage of the large amounts of visual data independent of their task-specific tooltips, while also using tooltip labels to seed the simulation of additional tooltips to improve the model generalizability.

6.3.1 Learning Objectives

As introduced in Section 6.2, the learning objective is to learn a task-specific model ϕ that produces the tooltip transform $f = \phi(I_o)$. This model is trained over an offline dataset

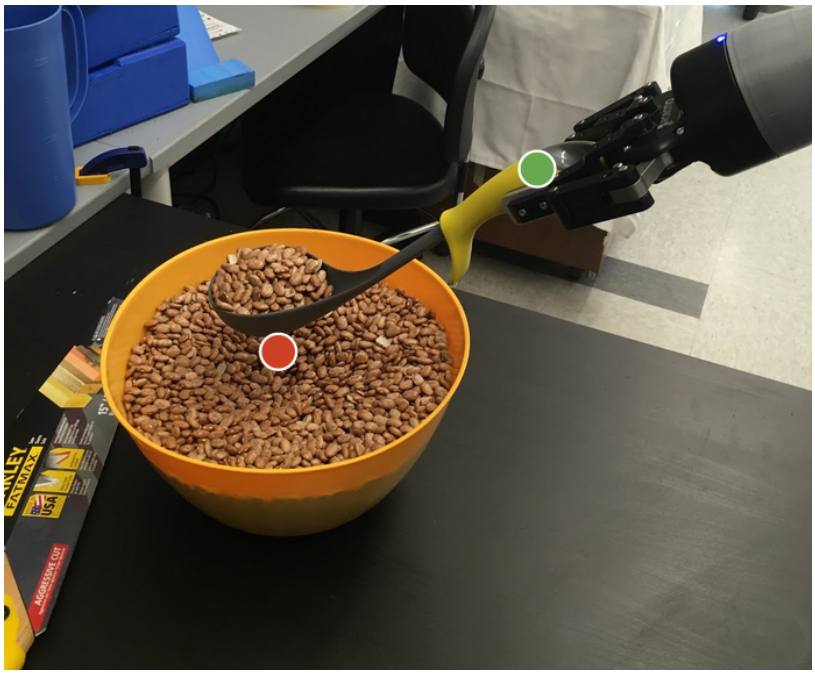


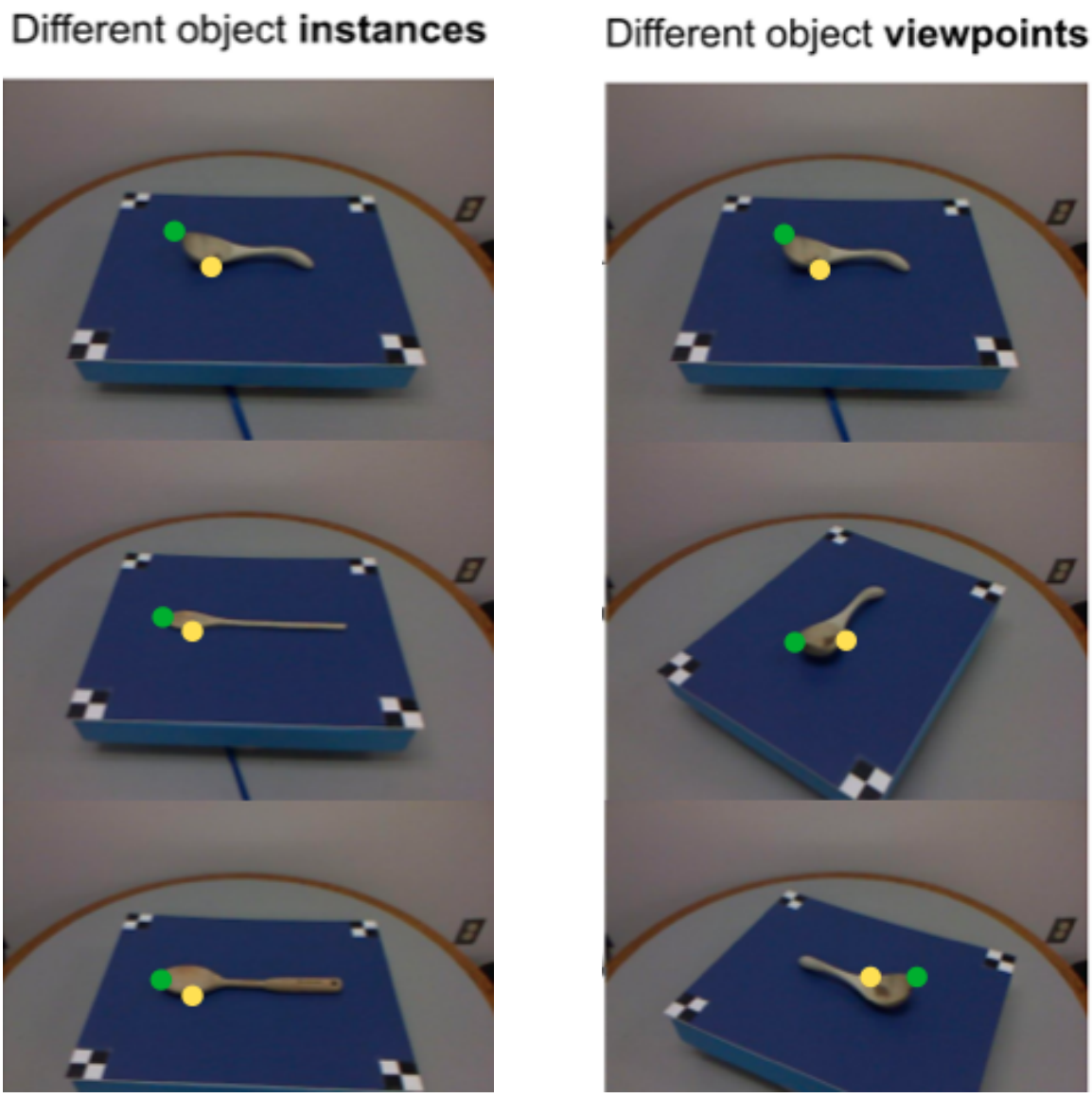
Figure 6.5: The tooltip position can be represented with respect to another part of the tool (e.g. the handle centered at the green dot) or with respect to the object it is used to manipulate (e.g. the scooping target centered at the red dot).

G_r and then tuned on demonstrations for a subset G_u of the new tool category. The tuned model is then tested on the remaining, undemonstrated tools G_e.

We consider two representations of the tooltip: a kinematic transform between the robot's gripper and the tooltip, and a pixel-wise tooltip selected within the robot's perception of the scene. While our goal is to ultimately obtain the tooltip transform, we consider both representations in order to leverage both large, offline image datasets and smaller, online kinesthetic demonstrations. We reformulate this learning objective in terms of both representations as follows:

$$f = \gamma(P_{o,a}) \qquad P_{o,a} = \phi_a(I_o)$$

Where γ is a robot-specific grounding function that translates a pixel-wise tooltip into a kinematic transform, and ϕ_a is a task-specific, pixel-wise tooltip predictor. We next discuss the training procedure for each objective.



(a) (b)

Figure 6.6: Examples of tooltips (shown in yellow and green) being projected consistently across different instances of a tool category as shown in (a), or across different viewpoints of a single tool as shown in (b).

6.3.2 Learning to Predict Candidate Tooltips

The first learning objective involves training a model ϕ that represents the visual features of a tool that are relevant to predicting its tooltips over a range of tasks. In offline training, we aim to train ϕ such that when it is tuned over a new category of tools, it produces a feature representation of each tool that is linearly related to the pixel-wise tooltip position.

To address the task-specific nature of tooltips, the training dataset must contain multiple tooltips for each tool. As discussed in Section 6.1, however, ground-truth tooltip data is not obtainable from a dataset since it is task-specific. We propose that, rather than use ground-truth tooltip data, the model may be trained over a set of *candidate* tooltips. This may be achieved by simulating a randomly generated tooltip for all tools within each training batch, where a new tooltip is generated for each training batch. This tooltip may be seeded according using the tools' affordance labels as a heuristic (e.g. seeded based on the centroid or outer edges of an affordance region).

A key challenge of this approach is projecting a simulated tooltip *consistently* across different variations and viewpoints of a tool category. Figure 6.6 illustrates this challenge. In Figure 6.6a, two tooltips (the end of the scoop and the side edge of the scoop, shown in green and yellow respectively) are projected consistently across different instances of a scoop tool. Similarly, in Figure 6.6b, the same two tooltips are projected consistently onto different viewpoints of the same tool. Due to the visual differences in these images and their respective feature representations, a key challenge is identifying which features enable consistent projection of a single tooltip onto each image. We propose that this challenge of consistent tooltip projection may be addressed as a separate learning problem, or by identifying a warping function that most closely maps one tool image onto another to identify their corresponding candidate tooltips.

6.3.3 Grounding Pixel-wise Tooltips in a Kinematic Transform

In Chapter 5, we demonstrated how Transfer by Correction enables a robot to learn a tool-task transform from interaction. Just as corrections provided samples of the motion constraints imposed by tools, we expect that they can be used to sample the grounding model γ. Rather than attempt to learn the relationship between the pixel-wise and kinematic transform representations of the tooltip, we assume that this relationship is linear and can be grounded during online tuning on the physical robot. During online tuning, the robot would receive corrections of a known task using a new tool, derive the tool-task transform as described in Chapter 5, and use the resulting datapoint (featurized image representation as input and tool-task transform as output) to ground a linear relationship between the expected pixel-wise tooltip and the corresponding kinematic transform.

6.3.4 Model Specifications

As discussed in the previous section, we optimize for two learning objectives: (1) training ϕ such that it outputs a pixel-wise tooltip representation based on visual features, and (2) training γ such that it produces the relationship between the pixel-wise tooltip representation and the kinematic transform between the robot's end-effector and the tooltip. This second objective may be addressed via corrections (as demonstrated in Chapter 5). We assume that this kinematic transform is linearly related to the pixel-wise tooltip position, and as a result, we focus on learning the pixel-wise representation ϕ as the primary learning objective for meta-learning. We propose using depth images as input due to (i) its availability from a standard RGB-D sensor and (ii) its emphasis on 3D structural features of the tools. Given a set of tool depth images as input, the model should return a representation of the tool image that is linearly related to the pixel-wise tooltip position and, by extension, is also linearly related to the kinematic transform.

We first propose deriving a 3D surface normal representation from the depth image, creating a combined 4D input (surface normals in 3D and depth in 1D). Similar to other

Algorithm 3 Interactive Tool-Task Grounding

Require: Learning rate parameters α and β

Initialize θ
for all training epochs **do**
    $l' \leftarrow 0$
    Sample batch of tool classes B_i
    for $b_i = (I_u, I_e) \in B_i$ **do**
        Generate tooltip t for a randomly selected image $i \in b_i$
        Let y_u be the projection of t onto all tuning images I_u in b_i
        Let y_e be the projection of t onto all testing images I_e in b_i
        $\theta' \leftarrow \theta$
        for each update step **do**
            $l \leftarrow loss_{l2}(\phi_{\theta'}(I_u), y_u)$
            Compute gradient $\nabla_{\theta'}$ from l
            $\theta' \leftarrow \theta' - \alpha \nabla_{\theta'}$
        $l' \leftarrow l' + loss_{l2}(\phi_{\theta'}(I_e), y_e)$
    Compute gradient ∇_θ from l'
    $\theta \leftarrow \theta - \beta \nabla_\theta$
return θ

regression and classification models that operate over image input, we propose using a convolutional neural network in which each convolutional layer is followed by batch normalization and a ReLU non-linearity. The last convolutional layer may produce a grid representing a downsampled, featurized representation of the input image, with each grid cell containing the probability that the corresponding image segment contains the tooltip. A final linear layer would then produce a 2D tooltip representation, to which an L2 loss function would be applied.

6.3.5 Algorithm

We adapt the algorithm structure used in the MAML algorithm [38], during which the model parameters are evaluated not on their immediate generalizability, but on their use as initial parameters when tuning the model for a new task. The training data is segmented to reflect this; each training iteration samples a tuning and query set from the same task, tunes the current model parameters on the tuning set, and then updates the model parameters based on the tuned parameters' performance on the query set.

6.4 Proposed Evaluation

We propose using images and affordance labels from the UMD Part Affordance Dataset [93] during offline training. This dataset contains RGB-D images for 105 tools, grouped into 17 object categories. Each tool is photographed at roughly 75 orientations, each of which corresponds to a pixel-wise labeling according to 7 possible affordances (e.g. cutting, grasping, pounding). We propose selecting a different affordance to seed the tooltip generator during each training iteration.

We first propose a simulated evaluation to test the model's ability to learn the pixel-wise tooltip representation ϕ. As a result, this evaluation focuses on identifying the object features relevant to identifying the tooltip, and does not address grounding that tooltip in its transform from the robot's gripper.

Our second proposed evaluation consists of tuning the meta-learned model over each of three real-world tool categories, consisting of three saws, three spatulas, and four ladles. The model would be tuned on two tasks per tool category; for example: aligning a saw, hanging a saw on a pegboard, prying with a spatula, flipping food with a spatula, scooping with a ladle, and hanging a ladle on a peg. For each tool-task combination, the learned model will be tuned using the transform demonstrated on that tool, after which, the tuned model's prediction of the tooltip would be evaluated for the remaining, unseen target tools within that same category.

We propose three baselines:

- Ground Truth: For each tool, compare the predicted transform to the demonstrated transform.
- Untransformed Demonstrations: Measure performance when directly reusing one tool's demonstrated transform on the other remaining tools within the same class.
- Object Scaling: Adapt the original task model based solely on the proportional difference between the original and new tools' bounding box dimensions.

We propose two performance metrics, collected over a range of training dataset sizes. These datasets may range according to the number of tools used to tune the model, and the number of images collected for each tuning tool. We expect that changing the tuning dataset along these two dimensions will affect the following performance metrics:

- A binary measure of task success, aggregated over all combinations of demonstration tool and target tool.
- Distance between predicted tooltip and ground truth tooltip (obtained via demonstrations).

6.5 Conclusion

We have considered two aspects of novelty that a robot may encounter in regards to tools: novel categories of tools, and novel variations of a tool. In order for a robot to adapt its task models to be used with a new category of tools, it must also learn to adapt to variations within that category as well. Recent advances in one-shot and meta learning are promising approaches to this challenge, but require extensive training datasets that are not feasibly obtainable for the task-specific nature of tool usage.

Similar to earlier chapters in this dissertation, we have first formalized the problem of how a robot can learn to generalize across different categories of tools, and then analyzed the data requirements necessary to enable meta-learning in this domain. We have proposed leveraging two sources of information: extensive, unlabeled datasets containing images of various tool categories, and later, a limited number of interactive demonstrations or assistance that are used to ground a kinematic representation for the visual model.

In order to implement this approach, the robot must have the ability to project simulated tooltips onto various tool images in a manner that is consistent over both (i) different variations of a tool, and (ii) different viewpoints of a tool variation. We have presented this as a challenge for future work; once addressed, we expect that meta-learning may then be

applied to the problem of tool transfer *without* the extensive training data requirements that are typical of this approach.

CHAPTER 7
CONCLUSION

Adaptability is an essential skill in human cognition, enabling us to draw from our extensive, life-long experiences with various objects and tasks in order to address novel problems. To date, robots do not have this kind of adaptability, and yet, as our expectations of robots' interactive and assistive capacity grows, it will be increasingly important for them to adapt to unpredictable environments in a similar manner as humans.

While a robot can be pre-programmed for many tasks and their variations, specifying these behaviors would require tedious effort, and still would not adequately prepare a robot for every scenario it may encounter. This is due to the various dimensions along which the source and target task may differ, such as changes in the task goals, task objects, manipulation tools, task constraints, and task dynamics. Existing work aims to generalize across variations in one of these dimensions by having the robot continue to explore its new environment, or by having a human teacher provide enough demonstrations to cover the space of possible task variations. All of these approaches require many examples of successful task completion across various environments, and do not consider how the quantity and environment setting of its training data affects the complexity of the transfer problem.

This dissertation is the first to present an interactive paradigm for task transfer. Rather than require more demonstration data in order to attempt generalization across these contexts, this dissertation instead leverages continued interaction with the teacher within the context of the target task. This enables a robot to directly model the relationship between environment changes and task adaptations. Furthermore, this allows the robot to quickly learn the salient task differences for transfer to a specific environment, regardless of the number or variations of previous demonstrations. This dissertation presents four primary contributions, summarized in Figure 7.1 and in the following section.

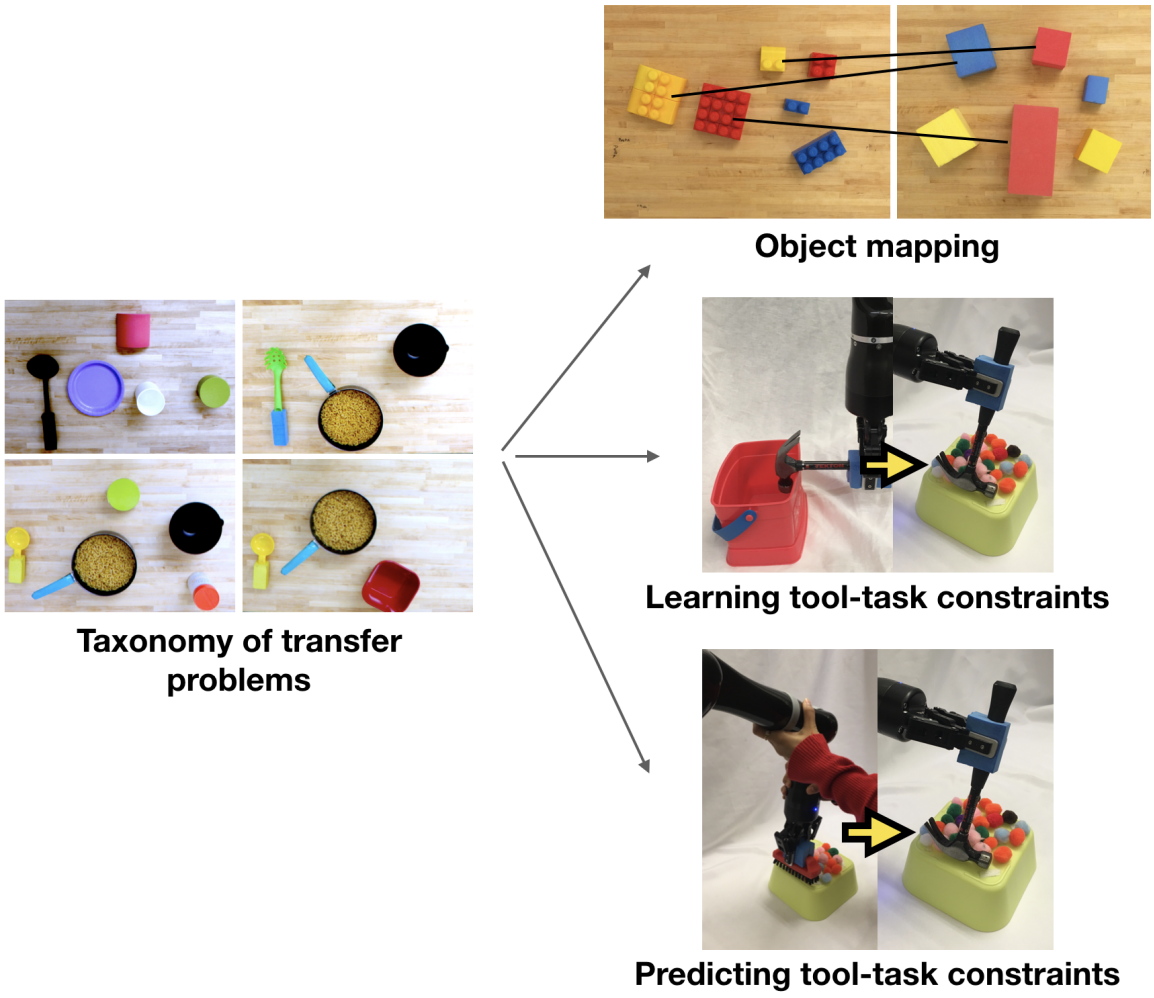


Figure 7.1: Thesis structure: we have presented a taxonomy of transfer problems, from which, we have addressed three particular categories of task transfer.

7.1 Summary of Contributions

7.1.1 Taxonomy of Transfer Problems

Contribution 1: A taxonomy of transfer problems, forming a relationship between (i) source and target task similarity, (ii) the level of abstraction at which the source task is represented, and (iii) an interactive method for grounding the abstracted task representation in the target domain.

Contribution 2: The Tiered Task Abstraction: a task representation that supports abstraction to address a range of task variations.

Summary: In Chapter 3, we defined a spectrum of task similarity according to the complexity of the state space mapping between two tasks. We addressed transfer problems that are solvable by identifying a mapping between the robot's state space in the learned, source task and the corresponding states in the new, target task, where the state space consists of the position of objects in the robot's environment and its end-effector position. We proposed that the difference between source and target environments dictates (i) the complexity of the state space mapping between the two tasks, (ii) the level of abstraction at which the task representation should be transferred and (iii) the dimensionality of the information needed to ground that representation for the target problem. We defined the Tiered Task Abstraction (TTA) as a task representation that enables abstraction of the task model into action models, parameterization functions, feature selectors, and feature values. Throughout the remainder of this dissertation, we evaluated this hypothesized relationship by analyzing three categories of transfer tasks spanning the range of state space mapping complexity: mappings between objects in the robot's source and target environments, mappings between objects and the robot's end-effector pose (e.g. a static robot pose transformation), and mappings between objects and the robot's end-effector pose with respect to object features (e.g. a robot pose transformation that is parameterized based on

object features).

Results and Key Insights: We applied three abstractions of the TTA representation to three categories of environment variations for two tasks. Our results indicate that as the source and target environments become more dissimilar, the task representation must be abstracted accordingly in order to successfully execute the task in the target environment. The most-abstracted task representation achieved the highest performance in all task variations. This demonstrates (i) a **correlation between task similarity, abstraction, and transfer performance**, and (ii) a **tradeoff between the generality of a task representation** (e.g. the range of transfer problems that it can successfully address) **and the data requirements that must be met to ground the abstracted task** representation in the target environment. This motivates the need for transfer methods that target a *specific* type of task variation in order to (i) select the appropriate level of abstraction and (ii) minimize the data required to ground the abstracted representation in the target environment.

7.1.2 Mapping by Demonstration

Contribution 3: Mapping by Demonstration: an interactive approach to confident, task-dependent object mapping (*"situated mapping"*) problems.

Summary: In Chapter 4, we addressed transfer problems that require a particular category of state space mappings: mappings between objects. This chapter addresses transfer problems in which the source and target environments differ in the objects used to complete the task. The contextual nature of object usage is a primary challenge of this problem; the objects the robot should use (and the order which they are used) is dependent on the task goals, subgoals, and how objects are used to fulfill those goals. Prior work assumes (i) the robot has access to multiple new demonstrations of the task or (ii) the primary features for object mapping have been specified. We introduce the *Mapping by Demonstration* algorithm, which is not constrained by either assumption, but rather, uses structured interaction

with a human teacher to infer an object mapping for task transfer.

After assisting the robot with the initial steps of a task in the target environment, the robot may then complete the rest of the task autonomously once it has inferred a correct mapping. Overall, our contributions in Chapter 4 enable a robot to transfer a known task to an environment containing new objects, operating under the assumption that the new objects are manipulated in the same way as those in the source environment.

Results and Key Insights: We first performed an extensive simulated evaluation, where the robot received mapping assistance from an oracle. Our simulated results indicate that the **problem complexity increases exponentially with the number of objects in the environment**, and that the **effect of this increase in complexity is minimized using the Mapping by Demonstration** algorithm.

We performed an additional, interactive evaluation to test how the algorithm responds to human-provided assistance. In this evaluation, user study participants demonstrated several tasks and then provided mapping assistance for each task using a target set of objects. We found that human-provided assistance resulted in noisy assistance data, either from a mis-alignment of the robot's and teacher's model of the task, or from mis-recordings of the teacher's assistance. Following this user study, we presented a confidence metric to account for noisy assistance data by moderating the number of assists requested by a robot. The resulting Confident Mapping by Demonstration approach maximizes the robot's autonomy by reducing the amount of required assistance, while also maximizing the robot's correctness during object mapping by reducing the opportunity for noisy assistance.

Our results indicate that human-guided object mapping provided a balance between mapping performance and autonomy, resulting in (i) **up to 2.25x as many correct object mappings as mapping without human interaction**, and (ii) **more efficient transfer** than requiring the human teacher to re-demonstrate the task in the new environment, **correctly inferring the object mapping across 93.3% of the tasks** and requiring at most one inter-

active assist in the typical case.

7.1.3 Transfer by Correction

Contribution 4: Transfer by Correction: a novel, interactive approach for grounding and modeling the end-effector transform used to manipulate a new tool.

Summary: In Chapter 5, we addressed transfer problems in which object replacements necessitate a mapping between the robot's end-effector when manipulating the original and new objects. We focused on tool replacement as a prototypical example of this category of transfer problems; two tools may be used to complete the same task, while also being comprised of different grasps, tooltips, and geometry that affect the optimal end-effector pose used to complete the task.

We introduced *Transfer by Correction*: a method for modeling and learning the end-effector transform that enables a robot to adapt a known task model to use a new tool. Our approach involves the robot receiving corrections from a human teacher when repeating a known task with a new tool. These corrections provide (noisy) samples of the transform between two tools, which we model in order to generate a typical tool transform. We contribute two models for representing corrections, a linear and rotation model, that each reflect a different relationship between the end-effector's position and orientation when using a tool. We have also presented a metric for choosing the better-fitting model for a set of corrections.

Results and Key Insights: In our within-task evaluation, we have demonstrated that the **linear and rotational models effectively represent corrections** for successful task completion with the new tool **in 83% of task executions**. Furthermore, using a metric to select the **best-fitting model resulted in improved performance in 89% of tasks** (in comparison to the original, untransformed trajectory).

We further evaluated whether the transforms learned in the context of one task may also

generalize to other tasks using the same tool replacement. When successful, this results in transfer without requiring any demonstrations or corrections for that tool-task combination. We found that the transforms learned from a single round of **corrections generalize to unseen tool/task combinations in 27.8% of our transfer evaluations, and up to 41% of transfer problems** when the source and replacement tool share tooltip similarities. Overall, these results indicate that successful task adaptation for a new tool is dependent on the the tool's usage within that task, and that the transform model learned from interactive corrections can be generalized to other tasks providing a similar context for the new tool.

7.2 Open Questions

This dissertation emphasizes the importance of adapting to the *type* of novelty encountered in a target task, and how the type of novelty dictates the task knowledge (and abstraction of that knowledge) needed to address the target task. We have presented and evaluated solutions to transfer problems requiring a mapping between source and target environment objects (Mapping by Demonstration) or the robot's end-effector poses (Transfer by Correction). In Chapter 6, we have also proposed a method for learning a parameterized relationship between environment object features and the robot's end-effector pose. We expect that by addressing the challenge of consistent tooltip projection (described in Section 6.3.2) in future work, meta-learning may be successfully applied to across-tool transfer tasks on a physical robot. Furthermore, we propose four additional opportunities for future research.

7.2.1 Assessing Tools for Improvisation

This dissertation is inspired by processes of human cognition; namely, the ability that humans have to quickly adapt to novelty in our environments. An extension of this cognitive ability is improvisation, in which an agent achieves a goal using atypical means. For a robot that learns and transfers manipulation tasks, improvisation would involve the robot completing a known task using an atypical approach or tool that was not demonstrated to it

by a human teacher.

In Chapter 5, we presented Transfer by Correction and demonstrated how this method may be used to perform a task using an atypical tool replacement. This work has shown it is possible for a robot to use non-canonical tools for a task. However, it relies on a key assumption: that the replacement tool provided to the robot is known to be capable of performing that task. This assumption is reasonable for a transfer problem in which a replacement tool is provided to the robot, but is not ideal for a robot that improvises and needs to select a tool that is appropriate for the task.

This challenge presents an opportunity for future work: learning the qualities of a tool that enable it to be used for a particular task, and then using this knowledge to assess the suitability of various tools in the context of that task. As an example, when applied to the sweeping task discussed in Chapter 5, this would involve the robot first learning about the qualities of a tool that enable sweeping: namely, that the tool must contain a surface that (i) is approximately the same width as the target surface to be swept and (ii) can be aligned parallel to the target surface. Second, the robot would need to identify candidate tooltips for each candidate tool, after which it would evaluate the most suitable tool and corresponding tooltip based on the tool qualities required by the task. Finally, the robot would need to adapt its task model to accommodate the relationship between the robot's end-effector and the selected tooltip.

7.2.2 Transfer to Novel State and Action Representations

The transfer problems we address in this dissertation are solved by identifying a state space mapping via interaction. This prompts additional research into transfer problems that are *not* addressed using a state space mapping. For example, this dissertation assumes that the state space representation itself remains static. For a robot that requires a new state space representation for a particular task, it would need to learn a mapping between corresponding state space elements as well.

Beyond changes to the robot's state space, a new task may invoke changes in its action space as well. Changes to the robot's kinematics may cause its learned task models to produce a different effect than what was originally learned, and thus it must adapt its task model accordingly. While adapting to new action spaces has been addressed in the reinforcement learning literature, guided interaction may provide additional benefits. For example, the data collected from interactive corrections (such as the data collected and modeled in Transfer by Correction, Chapter 5) could be extended to modeling action-space mappings as well.

7.2.3 Assessing Task Similarity Over Time

In Chapter 3, we demonstrated the relationship between task similarity and the level of abstraction at which transfer can successfully occur. Following this, in Chapters 4-6 we have shown that when given an abstracted task representation, the robot can successfully ground this representation using data collected from interaction. In total, once the similarity between a source and target environment has been established, the work presented in this dissertation enables a robot to abstract and ground the task representation accordingly.

While we have defined the features relevant to classifying the similarity between a source and target environment, it is a different matter for the robot to detect these features autonomously. Furthermore, we have assumed that the robot assesses the similarity between source and target environments a priori, and then utilizes the corresponding level of abstraction for the entirety of the task. Rather than utilize a static abstraction for the full task, there is an opportunity for future research to consider a dynamic task abstraction that is updated at each step of the task. This would enable the robot to select the appropriate task representation (and subsequent interaction for grounding) based on the task differences that are relevant to a single step of the task. In doing so, the robot may tailor the frequency and method of interaction it requests so as to (i) model the exact information needed for transfer, and (ii) maximize its overall autonomy.

7.2.4 Learning Task Goals to Facilitate Transfer

Finally, this dissertation may be augmented by incorporating reasoning over task goals during transfer. Task goals are typically used to impose a top-down constraint on the robot's actions. If the task goals are known by the robot, it could be used to greatly improve task transfer via self-supervision; given a model of the task goals (and thus its success and failure conditions), the robot may refine its task model in simulation until it has transferred the task model correctly. Additionally, having a model of the task goals may enable it to perform transfer over the goal model *itself*; e.g. learning to predict the goal for a new task.

However, task goals or constraints are often hand-coded by a human, as this information is difficult to learn through limited observations or demonstrations. As hand-coded data is not scalable to novel tasks, a primary challenge of using task goals to perform transfer is deriving the goal information. One potential source of this goal information is via interaction with the human teacher. While the teacher may describe the task goal verbally, they are unlikely to describe the goal in a manner that can be directly mapped to the robot's representation of the goal. Other forms of interaction may be used to indicate the task goal, such as providing demonstrations of successful or unsuccessful goal states, or by indicating features of the state representation that are relevant to the task goal. Additionally, active learning may be leveraged to enable the robot to request specific demonstrations or interactions based on its own goal representation. Future work may address this problem of deriving goal information from interaction such that the resulting goal representation can be used to guide task transfer.

Another source of goal information may result from task transfer itself. In this dissertation, we have considered a robot that performs transfer within the scope of a single task and target environment. However, for a robot in a long-term autonomy setting, it will likely learn many tasks that are transferred to many environment variations. Ideally, a robot that learns to transfer a task to a particular environment should retain the knowledge obtained during transfer such that it can be reused when transferring the task to another target envi-

ronment. As the robot transfers the task to multiple environments over time, it may collect enough data to learn a hypothesis of the task goal. Future work may address this challenging of leveraging data collected during transfer such that it enables lifelong learning.

7.2.5 Closing Thoughts

As social robots become more prevalent in society, it becomes increasingly important that they can adapt to and learn from humans in their environment. We envision a future of collaborative, adaptive robots that continuously learn from both explicit data (such as sensor readings and demonstrations) and implicit data (such as the commonsense reasoning implied by a teacher's demonstrations or corrections). This dissertation makes important steps toward achieving this goal; we have shown that structured interaction between a robot and a human teacher enables the robot to model (i) changes in its environment and (ii) how these changes affect its ability to complete tasks. We have proposed four additional avenues for future work that further advance this research vision, either by broadening the scope of novel situations that the robot may address (Sections 7.2.2 and 7.2.4) or by considering how a robot may leverage its experiences over time to more effectively address novelty (Sections 7.2.1 and 7.2.3). The technical contributions made by this dissertation provide a foundation for these future research directions, building toward a future of collaborative, adaptive robots.

REFERENCES

[1] S. Šabanović, "Robots in society, society in robots," *International Journal of Social Robotics*, vol. 2, no. 4, pp. 439–450, 2010.

[2] A. Tapus, M. Maja, and B. Scassellatti, "The grand challenges in socially assistive robotics," *IEEE Robotics and Automation Magazine, Institute of Electrical and Electronics Engineers*, 2007.

[3] D. Feil-Seifer and M. J. Mataric, "Defining socially assistive robotics," in *International Conference on Rehabilitation Robotics (ICORR)*, IEEE, 2005, pp. 465–468.

[4] T. Belpaeme, J. Kennedy, A. Ramachandran, B. Scassellati, and F. Tanaka, "Social robots for education: A review," *Science Robotics*, vol. 3, no. 21, 2018.

[5] B. KUKA Roboter GmbH. (2005). Factory automation robotics palettizing bread. `https://upload.wikimedia.org/wikipedia/commons/2/22/Factory_Automation_Robotics_Palettizing_Bread.jpg`.

[6] S. Jurvetson. (2012). Tesla robot dance. `https://www.flickr.com/photos/jurvetson/7408451314`.

[7] S. Jurvetson. (2010). Wandering in Willow Garage. `https://www.flickr.com/photos/jurvetson/5147429282`.

[8] Emshin. (2006). Robot concierge. `https://www.flickr.com/photos/emshin/303671580/`.

[9] NASA. (2013). ISS-36 Chris Cassidy works with Robonaut 2. `https://spaceflight.nasa.gov/gallery/images/station/crew-36/hires/iss036e013170.jpg`.

[10] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[11] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.

[12] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1463–1467, 2008.

[13] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[14] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *ACM/IEEE International Conference on Human-Robot Interaction*, ACM, 2012, pp. 391–398.

[15] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the people: The role of humans in interactive machine learning," *AI Magazine*, vol. 35, no. 4, pp. 105–120, 2014.

[16] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 4019–4026.

[17] T. W. Killian, S. Daulton, G. Konidaris, and F. Doshi-Velez, "Robust and efficient transfer learning with hidden parameter markov decision processes," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6250–6261.

[18] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1087–1098.

[19] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *ACM/IEEE International Conference on Human-Robot Interaction*, ACM, 2012, pp. 17–24.

[20] D. Martínez, G. Alenyà, P. Jiménez, C. Torras, J. Rossmann, N. Wantia, E. E. Aksoy, S. Haller, and J. Piater, "Active learning of manipulation sequences," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 5671–5678.

[21] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters, "Active reward learning.," in *Robotics: Science and Systems*, 2014.

[22] B. D. Argall, E. L. Sauser, and A. G. Billard, "Tactile guidance for policy refinement and reuse," in *IEEE International Conference on Development and Learning (ICDL)*, IEEE, 2010, pp. 7–12.

[23] E. L. Sauser, B. D. Argall, G. Metta, and A. G. Billard, "Iterative learning of grasp adaptation through human corrections," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 55–71, 2012.

[24] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from physical human corrections, one feature at a time," in *ACM/IEEE International Conference on Human-Robot Interaction*, ACM, 2018, pp. 141–149.

[25] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*, Springer, 2006, pp. 261–280.

[26] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[27] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2009, pp. 763–768.

[28] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," in *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, ACM, 2007, p. 233.

[29] D. Bruno, S. Calinon, and D. G. Caldwell, "Learning adaptive movements from demonstration and self-guided exploration," in *Joint IEEE International Conferences on Development and Learning and Epigenetic Robotics (ICDL-Epirob)*, IEEE, 2014, pp. 101–106.

[30] G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters, "Active incremental learning of robot movement primitives," 2017.

[31] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *The Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[32] M. E. Taylor, S. Whiteson, and P. Stone, "Transfer via inter-task mappings in policy search reinforcement learning," in *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, ACM, 2007, p. 37.

[33] B. Hayes, E. C. Grigore, A. Litoiu, A. Ramachandran, and B. Scassellati, "A developmentally inspired transfer learning approach for predicting skill durations," in *International Conference on Development and Learning and on Epigenetic Robotics*, IEEE, 2014, pp. 181–186.

[34] P. Parashar, A. K. Goel, B. Sheneman, and H. I. Christensen, "Towards life-long adaptive agents: Using metareasoning for combining knowledge-based planning with situated learning," *The Knowledge Engineering Review*, vol. 33, 2018.

[35] A. Goel, T. Fitzgerald, and P. Parashar, "Analogy and metareasoning: Cognitive strategies for robot learning," in *Human-Machine Shared Contexts*, W. Lawless, R. Mittu, and D. Sofge, Eds., Elsevier, 2020, ch. 2, pp. 23–44.

[36] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks," in *International Conference on Machine Learning (ICML)*, 2018.

[37] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference on Robot Learning*, 2017, pp. 357–368.

[38] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, 2017, pp. 1126–1135.

[39] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in *Conference on Robot Learning*, 2018, pp. 617–629.

[40] H. Hu, L. Chen, B. Gong, and F. Sha, "Synthesize policies for transfer and adaptation across tasks and environments," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 1176–1185.

[41] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *IEEE International Conference on Development and Learning (ICDL)*, IEEE, 2008, pp. 91–96.

[42] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning task-oriented grasping for tool manipulation from simulated self-supervision," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, 2018.

[43] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," 2018.

[44] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *The Artificial Intelligence Review*, vol. 11, no. 1-5, p. 11, 1997.

[45] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 75–113, 1997.

[46] S. Schaal and C. G. Atkeson, "Robot juggling: Implementation of memory-based learning," *Control Systems, IEEE*, vol. 14, no. 1, pp. 57–71, 1994.

[47] J. Kolodner, *Case-based reasoning*. Morgan Kaufmann, 1993.

[48] B. Falkenhainer, K. D. Forbus, and D. Gentner, "The structure-mapping engine: Algorithm and examples," *Artificial Intelligence*, vol. 41, no. 1, pp. 1–63, 1989.

[49] P. Thagard, K. J. Holyoak, G. Nelson, and D. Gochfeld, "Analog retrieval by constraint satisfaction," *Artificial Intelligence*, vol. 46, no. 3, pp. 259–310, 1990.

[50] A. K. Goel, "Design, analogy, and creativity," *IEEE Expert*, vol. 12, no. 3, pp. 62–70, 1997.

[51] A. K. Goel and S. R. Bhatta, "Use of design patterns in analogy-based design," *Advanced Engineering Informatics*, vol. 18, no. 2, pp. 85–94, 2004.

[52] M. W. Floyd, B. Esfandiari, and K. Lam, "A case-based reasoning approach to imitating robocup players.," in *FLAIRS Conference*, 2008, pp. 251–256.

[53] P. Thagard, *Mind: Introduction to cognitive science*. MIT press, 2005.

[54] G. Fauconnier and M. Turner, *The way we think: Conceptual blending and the mind's hidden complexities*. Basic Books, 2008.

[55] A.-M. Olteţeanu and Z. Falomir, "Object replacement and object composition in a creative cognitive system. towards a computational solver of the alternative uses test," *Cognitive Systems Research*, vol. 39, pp. 15–32, 2016.

[56] S. Ontañón, K. Mishra, N. Sugandh, and A. Ram, "Case-based planning and execution for real-time strategy games," in *Case-Based Reasoning Research and Development*, Springer, 2007, pp. 164–178.

[57] H. W. Park and A. M. Howard, "Case-based reasoning for planning turn-taking strategy with a therapeutic robot playmate," in *IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, IEEE, 2010, pp. 40–45.

[58] M. W. Floyd and B. Esfandiari, "A case-based reasoning framework for developing agents using learning by observation," in *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2011, pp. 531–538.

[59] M. Minor, S. Montani, and J. A. Recio-García, "Editorial: Process-oriented case-based reasoning," *Information Systems*, vol. 40, pp. 103–105, 2014.

[60] J. Kendall-Morwick and D. Leake, "A study of two-phase retrieval for process-oriented case-based reasoning," in *Successful Case-based Reasoning Applications*, Springer, 2014, pp. 7–27.

[61] M. Minor and E. Schulte-Zurhausen, "Towards process-oriented cloud management with case-based reasoning," in *Case-Based Reasoning Research and Development*, Springer, 2014, pp. 305–314.

[62] S. Montani and G. Leonardi, "Retrieval and clustering for supporting business process adjustment and analysis," *Information Systems*, vol. 40, pp. 128–141, 2014.

[63] D. K. Misra, J. Sung, K. Lee, and A. Saxena, "Tell me Dave: Context-sensitive grounding of natural language to manipulation instructions," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 281–300, 2016.

[64] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, and Others, "A world wide web for robots," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[65] K. Bullard, B. Akgun, S. Chernova, and A. L. Thomaz, "Grounding action parameters from demonstration," in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016.

[66] M. Tenorth, D. Nyga, and M. Beetz, "Understanding and executing instructions for everyday manipulation tasks from the world wide web," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2010, pp. 1486–1491.

[67] J. Kulick, M. Toussaint, T. Lang, and M. Lopes, "Active learning for teaching a robot grounded relational symbols," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[68] C. Diuk, A. Cohen, and M. L. Littman, "An object-oriented representation for efficient reinforcement learning," in *International Conference on Machine Learning (ICML)*, ACM, 2008, pp. 240–247.

[69] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," in *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, New York, New York, USA: ACM Press, 2007, p. 1.

[70] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, pp. 1–25, 2009.

[71] A. X. Lee, A. Gupta, H. Lu, S. Levine, and P. Abbeel, "Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to de-

formable object manipulation," *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[72] S. H. Huang, J. Pan, G. Mulcaire, and P. Abbeel, "Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects," in *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[73] S. Brown and C. Sammut, "Tool use and learning in robots," in *Encyclopedia of the Sciences of Learning*, Springer, 2012, pp. 3327–3330.

[74] C. C. Kemp and A. Edsinger, "Robot manipulation of human tools: Autonomous detection and control of task relevant features," in *International Conference on Development and Learning (ICDL)*, vol. 42, 2006.

[75] H. Hoffmann, Z. Chen, D. Earl, D. Mitchell, B. Salemi, and J. Sinapov, "Adaptive robotic tool use under variable grasps," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 833–846, 2014.

[76] P. Gajewski, P. Ferreira, G. Bartels, C. Wang, F. Guerin, B. Indurkhya, M. Beetz, and B. Śniezyński, "Adapting everyday manipulation skills to varied scenarios," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 1345–1351.

[77] B. Akgun and A. Thomaz, "Simultaneously learning actions and goals from demonstration," *Autonomous Robots*, vol. 40, no. 2, pp. 211–227, 2016.

[78] S. Ekvall and D. Kragic, "Learning task models from multiple human demonstrations," in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2006, pp. 358–363.

[79] J. Kirk, A. Mininger, and J. Laird, "Learning task goals interactively with visual demonstrations," *Biologically Inspired Cognitive Architectures*, vol. 18, pp. 1–8, 2016.

[80] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[81] F. Fernández and M. Veloso, "Policy reuse for transfer learning across tasks with different state and action spaces," in *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[82] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning.," *Journal of Machine Learning Research*, vol. 8, no. 1, pp. 2125–2167, 2007.

[83] T. Fitzgerald, A. Goel, and A. Thomaz, "Human-robot co-creativity: Task transfer on a spectrum of similarity," in *International Conference on Computational Creativity (ICCC)*, 2017.

[84] A. J. B. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, "Efficient organized point cloud segmentation with connected components," *Semantic Perception Mapping and Exploration*, 2013.

[85] T. Fitzgerald, K. Bullard, A. Thomaz, and A. Goel, "Situated mapping for transfer learning," in *Conference on Advances in Cognitive Systems*, 2016.

[86] T. Fitzgerald, A. Goel, and A. Thomaz, "Human-guided object mapping for task transfer," *ACM Trans. Hum.-Robot Interact.*, vol. 7, no. 2, 17:1–17:24, Oct. 2018.

[87] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[88] T. Fitzgerald, E. Short, A. Goel, and A. Thomaz, "Human-guided trajectory adaptation for tool transfer," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1350–1358.

[89] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[90] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.

[91] J Traa, "Least-squares intersection of lines," *UIUC, Illinois*, 2013, `http://cal.cs.illinois.edu/~johannes/research/LS_line_intersect.pdf`.

[92] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, IEEE, 2015, pp. 928–935.

[93] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 1374–1381.